Red Hat Certificate System 9

管理指南

更新了 Red Hat Certificate System 9.7

Last Updated: 2023-03-25

Red Hat Certificate System 9 管理指南

更新了 Red Hat Certificate System 9.7

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本手册涵盖安装、配置和管理 &CRTS、子系统的所有方面。它还涵盖添加用户等管理任务；请求、更新和撤销证书；发布 CRL，以及管理智能卡。本指南适用于 &CRTS；管理员。

目录

第 1 章 RED HAT CERTIFICATE SYSTEMNBSP 概述;HAT CERTIFICATE RED HAT CERTIFICATE SYSTEMNBSP;SYSTEM SUBSYSTEMS

每个常见的 PKI 操作都 - 发布、更新和撤销证书；存档和恢复密钥；发布 CRL 并验证证书状态 - 由红帽证书系统设备中的交互子系统进行；红帽证书认证系统；验证证书；系统。本章介绍了各个子系统的功能和它们一起来构建强大且本地 PKI 的功能。

1.1. 使用证书

证书的目的是建立信任。它们的使用情况根据它们用于确保的信任类型而有所不同。某些种类的证书用于验证出发者的身份；另一些则用于验证对象或项目没有被篡改。

有关使用证书、证书类型或证书如何建立身份和关系的信息，请参阅 *红帽认证系统 9 规划、安装和部署指南中的证书和验证章节。* *https://access.redhat.com/documentation/en-US/Red_Hat_Certificate_System/9/html/Planning_Installation_and_Deployment_Guide/Introduction_to_Pu Certificates_and_Authentication.html#types-of-certificates*

1.2. CERTIFICATECERTIFICATE SYSTEMNBSP 的评论;系统 子系统

Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System 提供五个不同的子系统，它们分别侧重于 PKI 部署的不同方面。这些子系统共同创建公钥基础架构(PKI)。根据安装的子系统，PKI 可以充当令牌管理系统(TMS)或非令牌管理系统。有关子系统和 TMS 和非TMS 环境的描述，请参阅 红帽认证系统 规划、安装和部署指南中的证书系统 子系统 章节。

***Enterprise** 安全客户端*
企业安全客户端不是子系统，因为它不使用证书、密钥或令牌执行任何操作。Enterprise Security Client 是一个用户界面，它允许用户非常轻松地管理智能卡中的证书。Enterprise Security Client 会将所有令牌操作（如证书请求）发送到令牌处理系统(TPS)，然后将其发送到证书颁发机构(CA)。如需更多信息，请参阅 Red Hat Certificate System 9 使用 Enterprise Security Client 管理智能卡 。

1.3. 查看管理证书(NON-TMS)

传统的 PKI 环境提供了基本的框架，用于管理存储在软件数据库中的证书。这是一个 非TMS 环境，因为它不管理智能卡上的证书。至少，非TMS 只需要一个 CA，但非 RHEL 环境也可以使用 OCSP 响应器 和 KRA 实例。

有关此主题的详情，请参阅 红帽证书系统规划、安装和部署指南中的 以下章节：

- *管理证书*
- *使用单一证书管理器*
- *规划循环密钥：密钥存档和恢复*
- *平衡证书请求处理*
- *平衡客户端 OCSP 请求*

1.4. 查看令牌管理系统(TMS)

CertificateCertificate Systemnbsp;System 创建、管理、更新并撤销证书，同时存档和恢复密钥。对于使用智能卡的组织，证书Certificate Systemnbsp;System 有令牌管理系统 - 具有已建立关系的子系统集合 - 生成密钥和请求以及接收用于智能卡的证书。

有关此主题的详情，请参阅 红帽证书系统规划、安装和部署指南中的 以下章节：

- 使用智能卡(TMS)
- 使用智能卡

1.5. REDRED HAT CERTIFICATE SYSTEMNBSP;HAT CERTIFICATERED HAT CERTIFICATE SYSTEMNBSP;SYSTEM SERVICES

根据用户类型，管理证书和密钥有不同的接口：管理员、代理、审计员和最终用户。有关通过每个界面执行的不同功能的概述，请参阅 红帽认证系统 9 规划、安装和部署指南中的红帽认证系统 用户界面 部分。

部分 I. RED HAT CERTIFICATE SYSTEM USER INTERFACES

第 2 章 用户界面

根据用户的角色，管理证书和密钥有不同的接口：管理员、代理、审计员和最终用户。

2.1. 用户界面概述

管理员可以使用以下接口安全地与完成的证书系统安装交互：

- PKI 命令行界面和其他命令行实用程序
- PKI 控制台图形界面
- 证书系统 Web 界面.

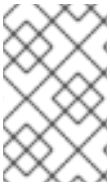
这些接口需要先配置，然后才能通过 TLS 与证书系统服务器进行安全通信。不允许在没有正确配置的情况下使用这些客户端。其中一些工具使用 TLS 客户端身份验证。如果需要，它们所需的初始化过程包括配置这一点。使用的接口取决于管理员的首选项和功能。本章会在指南的剩余部分中描述使用这些接口的常见操作。

默认情况下，PKI 命令行界面使用用户的 **~/.dogtag/** 目录中的 NSS 数据库。第 2.5.1.1 节 "pki CLI initialization" 提供使用管理员证书和密钥初始化 NSS 数据库的详细步骤。第 2.5.1.2 节 "Using "pki" CLI" 中介绍了使用 PKI 命令行工具的一些示例。其他示例将显示在本指南的其余部分中。

与证书系统（作为其他用户角色中的管理员）交互，可以使用各种命令行实用程序提交 CMC 请求，管理生成的证书等。这些在 第 2.5 节 "命令行界面" 中简单描述，如 第 2.5.2 节 "AtoB"。这些工具在稍后部分（如 第 5.2.1.2 节 "使用 **PKCS10Client** 创建 CSR"）中使用。

-- 证书系统 PKI 控制台界面是图形界面。第 2.3.1 节 "**pkiconsole** complete" 介绍如何初始化。第 2.3.2 节 "将 **pkiconsole** 用于 CA、OCSP、KRA 和 TKS Subsystems" 概述了使用控制台界面。后面部分，如 第 3.2.2 节 "使用基于 Java 的管理控制台管理证书注册配置集" 会根据特定操作进行更详细的信息。

证书系统 Web 界面允许通过 Firefox Web 浏览器访问。第 2.4.1 节 "浏览器初始化" 描述配置客户端身份验证的步骤。第 2.4 节 "Web 界面" 中的其他部分描述了使用证书系统的 Web 界面。



注意

要终止 PKI 控制台会话，请单击 **Exit** 按钮。要终止 Web 浏览器会话，请关闭浏览器。当命令行实用程序执行该操作并返回到提示时，命令行实用程序就会立即终止。因此，管理员部分无法终止会话。

2.2. 客户端 NSS 数据库初始化

在 Red Hat Certificate System 中，某些接口可能需要使用 TLS 客户端证书身份验证（验证）访问服务器。在执行服务器端的管理任务前，您需要：

1. 为客户端准备 NSS 数据库。这可以是新数据库或现有数据库。
2. 导入 CA 证书链并信任它们。
3. 具有证书和对应密钥。它们可以生成在 NSS 数据库中，或者从其他位置导入，比如从 PKCS #12 文件中导入。

根据 实用程序，您需要相应地初始化 NSS 数据库。请参阅：

- 第 2.5.1.1 节 "pki CLI initialization"

- 第 2.3.1 节 "**pkiconsole** complete"
- 第 2.4.1 节 "浏览器初始化"

2.3. 图形界面

pkiconsole 是一个图形界面，专为具有管理员角色特权的用户而设计，用于管理子系统本身。这包括添加用户、配置日志、管理配置文件和内部数据库，以及许多其他功能。此实用程序使用 client-authentication 通过 TLS 与证书系统服务器通信，并可用于远程管理服务器。

2.3.1. pkiconsole complete

要首次使用 **pkiconsole** 接口，请指定新密码并使用以下命令：

```
$ pki -c password -d ~/.redhat-idm-console client-init
```

这个命令会在 **~/.redhat-idm-console/** 目录中创建一个新的客户端 NSS 数据库。

要将 CA 证书导入到 PKI 客户端 NSS 数据库，请参阅 Red Hat 证书系统规划、安装和部署指南中的 将 证书导入到 NSS 数据库 部分。

要请求新的客户端证书，请参阅 第 5 章 请求、注册和管理证书。

执行以下命令从 **.p12** 文件中提取 admin 客户端证书：

```
$ openssl pkcs12 -in file -clcerts -nodes -nokeys -out file.crt
```

验证和导入 管理客户端证书，如 红帽认证系统规划、安装和部署指南中的 管理证书/密钥加密令牌 一节中所述：

```
$ PKICertImport -d ~/.redhat-idm-console -n "nickname" -t ",," -a -i file.crt -u C
```

重要

在导入 CA 管理客户端证书前，请确保已导入所有中间证书和 root CA 证书。

将现有客户端证书及其密钥导入到客户端 NSS 数据库中：

```
$ pki -c password -d ~/.redhat-idm-console pkcs12-import --pkcs12-file file --pkcs12-password
pkcs12-password
```

使用以下命令验证客户端证书：

```
$ certutil -V -u C -n "nickname" -d ~/.redhat-idm-console
```

2.3.2. 将 pkiconsole 用于 CA、OCSP、KRA 和 TKS Subsystems

Java 控制台由四个子系统使用：CA、OCSP、KRA 和 TKS。控制台可通过本地安装的 **pkiconsole** 工具访问。它可以访问任何子系统，因为该命令需要主机名、子系统的管理 TLS 端口和特定的子系统类型。

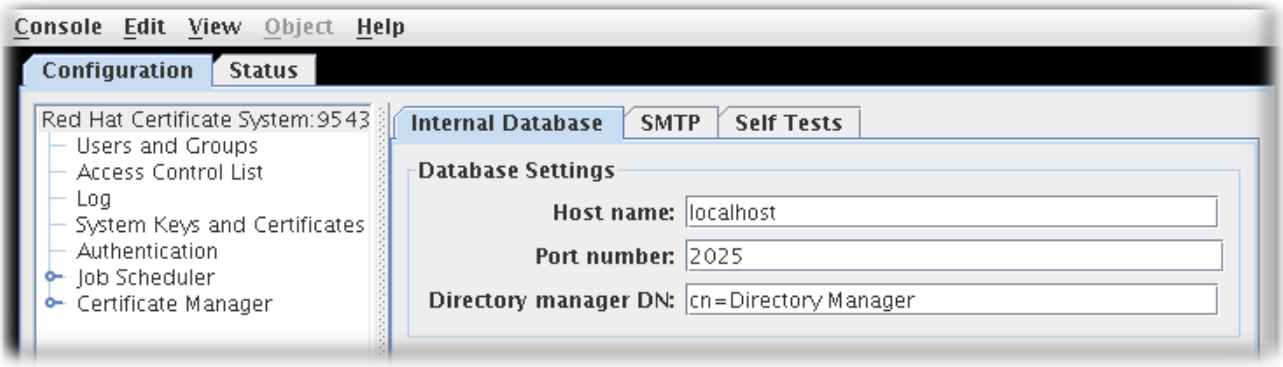
```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

如果没有配置 DNS，您可以使用 IPv4 或 IPv6 地址连接到控制台。例如：

> https://192.0.2.1:8443/ca
> https://[2001:DB8::1111]:8443/ca

这会打开一个控制台，如 图 2.1 "证书系统控制台" 所示。

图 2.1. 证书系统控制台



Configuration 选项卡控制子系统的所有设置，如名称所示。此选项卡中的选择根据实例的子系统类型而异；CA 具有最多选项，因为它具有适用于作业、通知和证书注册身份验证的其他配置。

所有子系统都有四个基本选项：

- 用户和组
- 访问控制列表
- 日志配置
- 子系统证书（根据子系统发布的证书，例如，在安全域或审计签名中）

Status 选项卡显示子系统维护的日志。

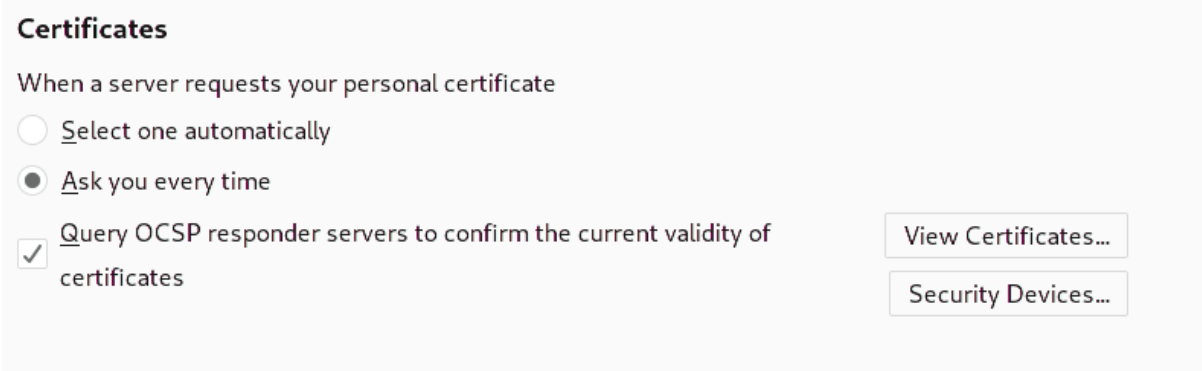
2.4. WEB 界面

2.4.1. 浏览器初始化

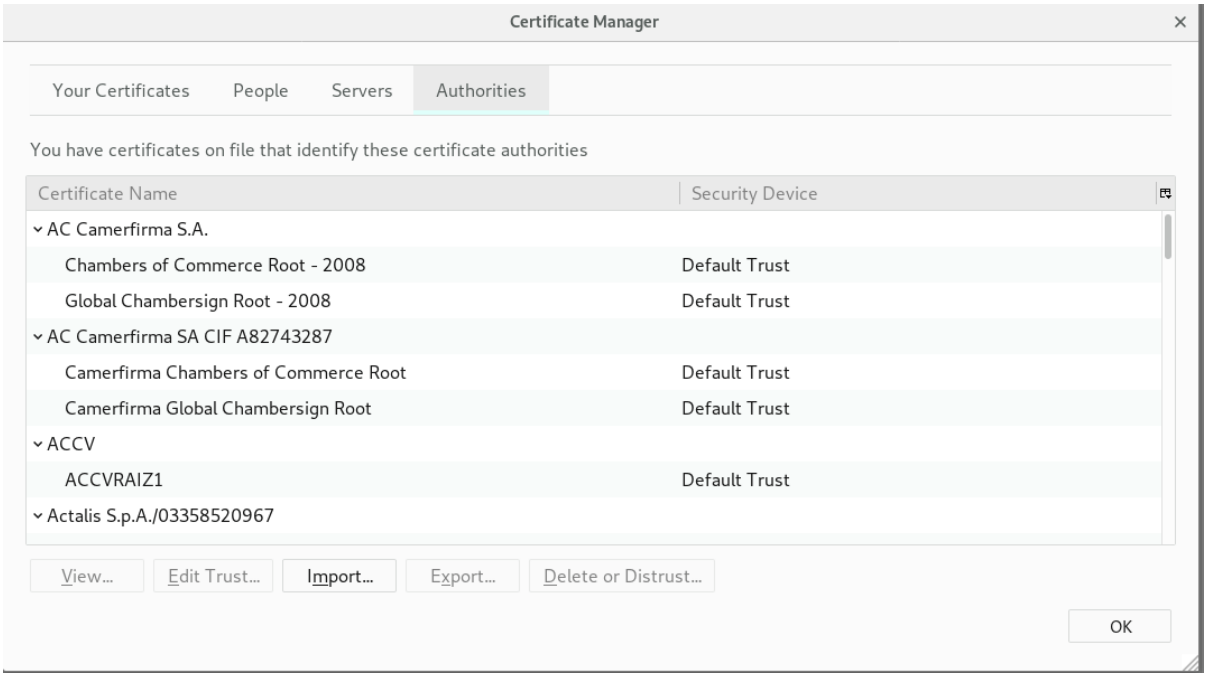
本节解释了 Firefox 访问 PKI 服务的浏览器初始化。

导入 CA 证书

1. 点击 **Menu → 首选项 → 隐私以及安全 →** 视图证书。



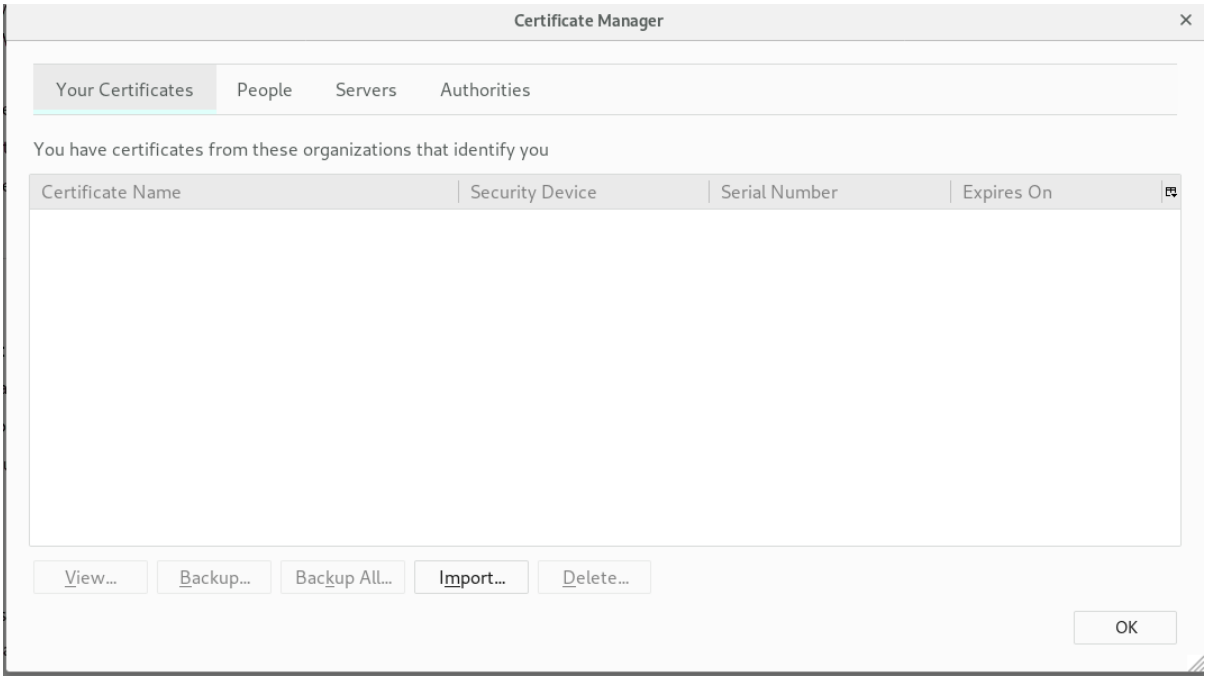
2. *选择"**颁发机构**"选项卡，然后单击"**导入**"按钮。*



3. *选择 **ca.crt** 文件并点击 **Import**。*

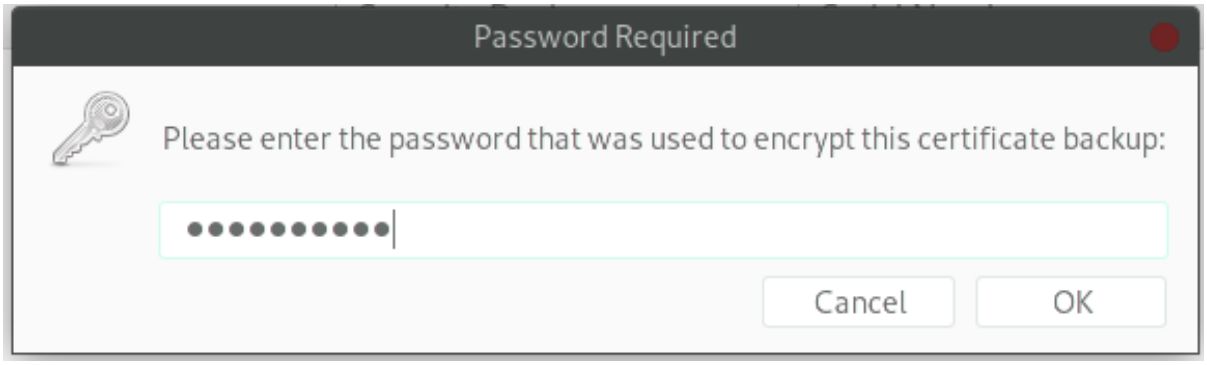
导入客户端证书

1. *点击 **选项→首选项→隐私以及安全→视图证书**。*
2. *选择您的 **证书** 选项卡。*



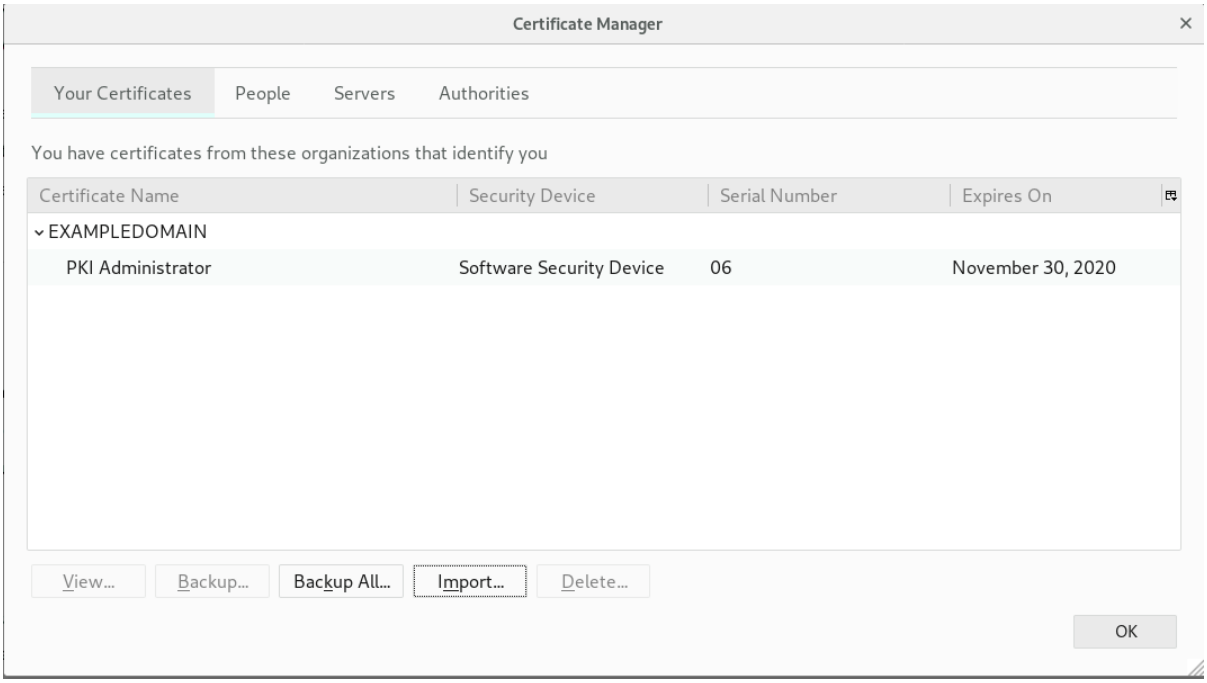
3. *单击* ***Import，再选择客户端 p12 文件，如 ca_admin_cert.p12。***

4. *在提示符下输入客户端证书的密码。*



5. *点击* ***OK。***

6. *验证您的证书下是否添加了* ***条目。***



*访问 **Web** 控制台*
*您可以通过在浏览器中打开 **https://host_name:端口** 来访问 PKI 服务。*

2.4.2. 管理接口

*所有子系统都使用基于 HTML 的管理界面。它通过输入主机名并以 URL 保护端口、与管理员的证书进行身份验证，然后单击相应的 **管理**员 链接。*

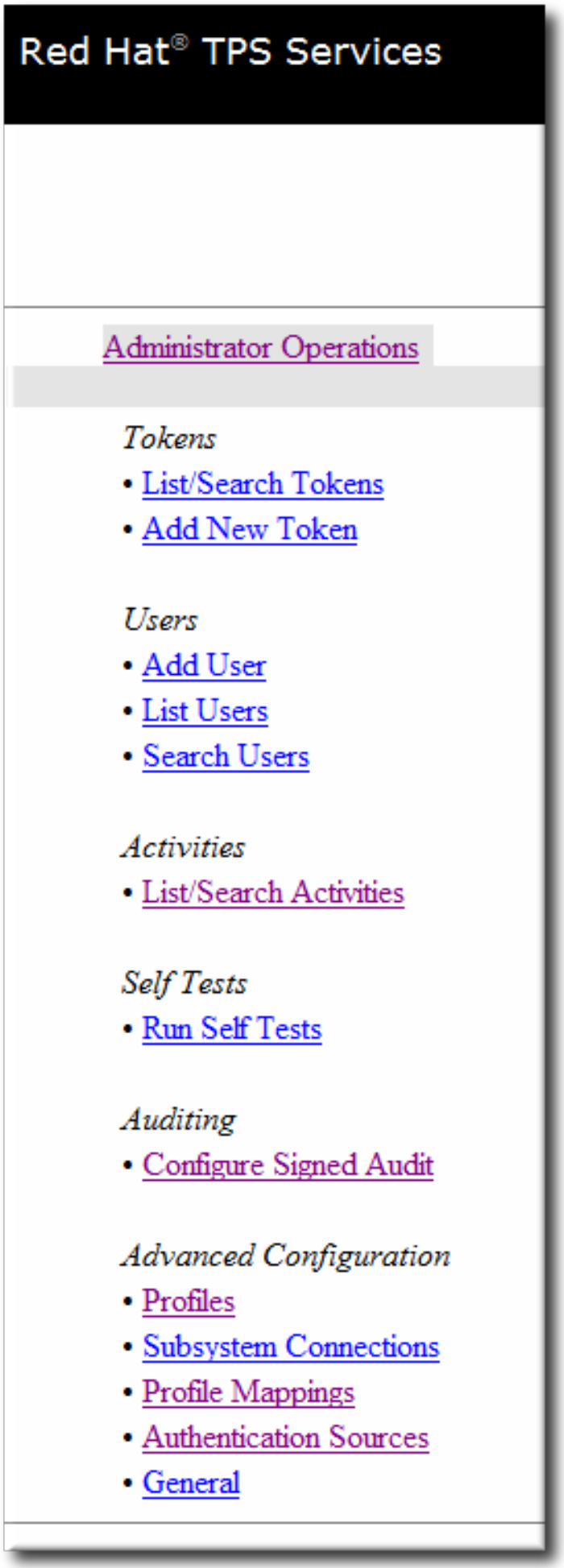
注意

所有子系统均只有一个 TLS 端口，它们用于管理员和代理服务。对这些服务的访问受基于证书的身份验证的限制。

HTML 管理界面比 Java 控制台更多；主要管理功能正在管理子系统用户。

TPS 仅允许操作来管理 TPS 子系统的用户。但是，TPS 管理员页面还可列出令牌，并显示在 TPS 上执行的所有活动（包括通常隐藏的管理操作）。

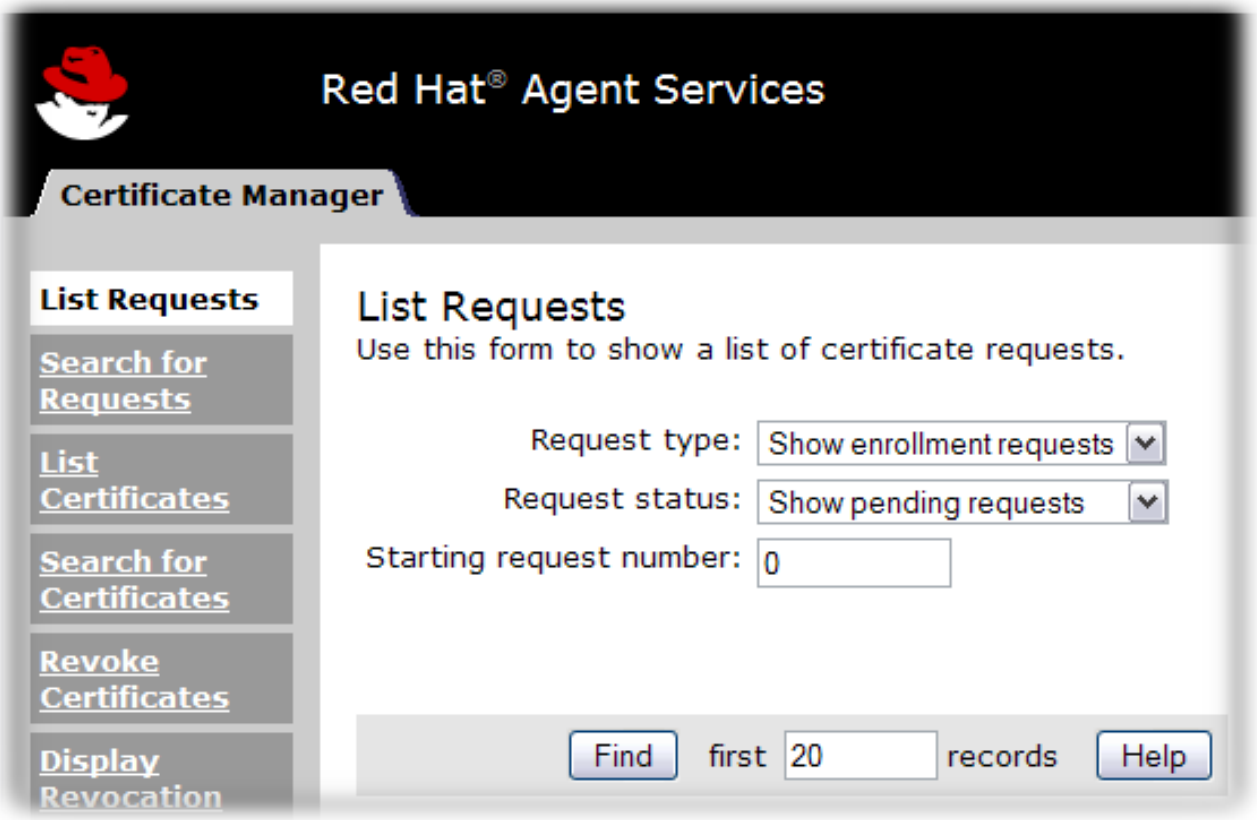
*图 2.2. **TPS 管理**员页**面***


Red Hat® TPS Services
Administrator Operations
Tokens
• List/Search Tokens
• Add New Token
Users
• Add User
• List Users
• Search Users
Activities
• List/Search Activities
Self Tests
• Run Self Tests
Auditing
• Configure Signed Audit
Advanced Configuration
• Profiles
• Subsystem Connections
• Profile Mappings
• Authentication Sources
• General

2.4.3. 代理接口

代理服务页面是几乎所有证书和密钥管理任务的执行位置。这些服务是基于 HTML 的，代理使用特殊的代理证书向站点进行身份验证。

图 2.3. 证书管理器的代理服务页面



具体操作会根据子系统的不同而有所不同：

- 证书管理器代理服务包括批准证书请求（发出证书）、撤销证书以及发布证书和 CRL。CA 发布的所有证书均可通过其代理服务页面进行管理。
- TPS 代理服务（如 CA 代理服务）管理已格式化的所有令牌，并通过 TPS 向它们发布证书。令牌可以被代理注册、暂停和删除。另外两个角色（operator 和 admin）可以查看 web 服务页面中的令牌，但不能对令牌执行任何操作。
- KRA 代理服务页面处理密钥恢复请求，这设定了在证书丢失时是否允许重新发布证书。
- OCSP 代理服务页面允许代理配置将 CRL 发布到 OCSP 的 CA，以手动将 CRL 加载到 OCSP，并查看客户端 OCSP 请求的状态。

TKS 是唯一没有代理服务页面的子系统。

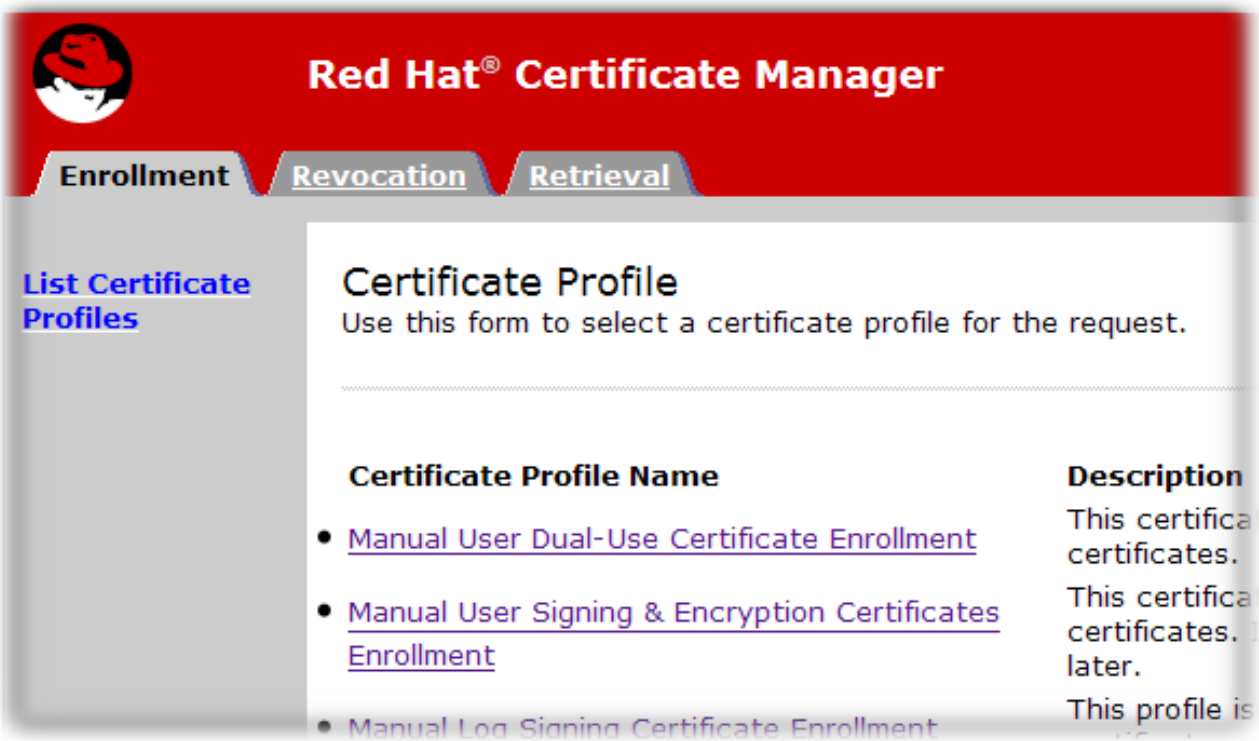
2.4.4. 最终用户页面

CA 和 TPS 均以某种方式处理直接用户请求。这意味着最终用户必须采用一种方法来连接这些子系统。CA 具有最终用户或 最终用户、 HTML 服务。TPS 使用企业安全客户端。

最终用户服务使用服务器的主机名和标准端口号通过标准 HTTP 访问；也可使用服务器的主机名和特定端点 TLS 端口通过 HTTPS 访问它们。

对于 CA，每种 TLS 证书类型通过特定的在线提交表单进行处理，称为 配置集。CA 有两个 dozen 证书配置文件，涵盖了所有种类的证书，包括用户 TLS 证书、服务器 TLS 证书、日志和文件签名证书、电子邮件证书以及每种子系统证书。还可能有自定义配置集。

图 2.4. 证书的 Manager 结束日期（Entities 页）



最终用户在签发证书时，通过 CA 页面检索其证书。它们也可以下载 CA 链和 CRL，并可以通过这些页面撤销或更新证书。

2.5. 命令行界面

本节讨论命令行实用程序。

2.5.1. "pki" CLI

pki 命令行界面(CLI)提供对使用 REST 界面的服务器各种服务的访问（请参阅 红帽认证系统规划、安装和部署指南中的 REST 接口 部分）。CLI 可以按照以下方式调用：

```
$ pki [CLI options] <command> [command parameters]
```

请注意，必须在命令前放置 CLI 选项，并在 命令之后放置命令参数。

2.5.1.1. pki CLI initialization

要首次使用命令行界面，请指定新密码并使用以下命令：

```
$ pki -c <password> client-init
```

这会在 **~/.dogtag/nssdb** 目录中创建一个新的客户端 NSS 数据库。必须在使用客户端 NSS 数据库的所有 CLI 操作中指定密码。或者，如果密码存储在文件中，您可以使用 **-C** 选项指定该文件。例如：

```
$ pki -C password_file client-init
```

要将 CA 证书导入到客户端 NSS 数据库，请参阅 红帽认证系统规划、安装和部署指南中的 将 证书导入到 NSS 数据库 部分。

有些命令可能需要客户端证书身份验证。要将现有客户端证书及其密钥导入到客户端 NSS 数据库中，请指定 PKCS #12 文件和密码，并执行以下命令：

执行以下命令从 **.p12** 文件中提取 admin 客户端证书：

```
$ openssl pkcs12 -in file -clcerts -nodes -nokeys -out file.crt
```

验证和导入 管理客户端证书，如 红帽认证系统规划、安装和部署指南中的 管理证书/密钥加密令牌 一节中所述：

```
$ PKICertImport -d ~/.dogtag/nssdb -n "nickname" -t ",," -a -i file.crt -u C
```

重要

在导入 CA 管理客户端证书前，请确保已导入所有中间证书和 root CA 证书。

要将现有客户端证书及其密钥导入到客户端 NSS 数据库中，请指定 PKCS #12 文件和密码，并执行以下命令：

```
$ pki -c <password> pkcs12-import --pkcs12-file <file> --pkcs12-password <password>
```

使用以下命令验证客户端证书：

```
certutil -V -u C -n "nickname" -d ~/.dogtag/nssdb
```

2.5.1.2. Using "pki" CLI

命令行界面支持以分级结构组织的多个命令。要列出顶级命令，在不执行任何附加命令或参数的情况下执行 **pki** 命令：

```
$ pki
```

些命令使用 子命令。要列出它们，请使用命令名称执行 **pki**，而不执行附加选项。例如：

```
$ pki ca
```

```
$ pki ca-cert
```

要查看命令用法信息，请使用 **--help** 选项：

```
$ pki --help
```

```
$ pki ca-cert-find --help
```

要查看 man page，请指定命令行 **帮助** 命令：

```
$ pki help
```

```
$ pki help ca-cert-find
```

要执行不需要身份验证的命令，请指定命令及其参数（如果需要），例如：

```
$ pki ca-cert-find
```

要执行需要客户端证书身份验证的命令，请指定证书 nickname、客户端 NSS 数据库密码以及可选的服务器 URL：

```
$ pki -U <server URL> -n <nickname> -c <password> <command> [command parameters]
```

例如：

```
$ pki -n jsmith -c password ca-user-find ...
```

默认情况下，CLI 与位于 **http://local_host_name 的服务器通信：8080**。要与位于不同位置的服务器通信，请使用 **-U** 选项指定 URL，例如：

```
$ pki -U https://server.example.com:8443 -n jsmith -c password ca-user-find
```

2.5.2. AtoB

AtoB 工具对 Base64 编码的证书进行解码，使其二进制文件等效。例如：

```
$ AtoB input.ascii output.bin
```

详情请查看 AtoB(1) man page。

2.5.3. AuditVerify

AuditVerify 程序通过验证日志条目中的签名来验证审计日志的完整性。

例如：

```
$ AuditVerify -d ~jsmith/auditVerifyDir -n Log Signing Certificate -a ~jsmith/auditVerifyDir/logListFile -P "" -v
```

这个示例使用 **~jsmith/auditVerifyDir** NSS 数据库中的 **Log Signing Certificate**（**-n**）验证审计日志。要验证（**-a**）的日志列表位于 **~jsmith/auditVerifyDir/logListFile** 文件中，以逗号分隔和排序。证书前的前缀（**-P**），密钥数据库文件名为空。输出是 verbose（**-v**）。

详情请查看 AuditVerify(1) man page 或 第 15.3.2 节 “使用签名审计日志”。

2.5.4. BtoA

BtoA 程序编码了 Base64 中的二进制数据。例如：

```
$ BtoA input.bin output.ascii
```

详情请查看 BtoA(1) man page。

2.5.5. CMCRequest

CMCRequest 程序创建一个证书保护或撤销请求。例如：

```
$ CMCRequest example.cfg
```

注意

CMCRequest 工具的所有选项都作为传递给实用程序的配置文件的一部分指定。有关配置文件选项和更多信息，请查看 CMCRequest(1) man page。另请参阅 4.3。使用 CMC 和 第 7.2.1 节 "使用 **CMCRequest**撤销证书" 请求和接收证书。

2.5.6. CMCRevoke

传统.不要使用。

2.5.7. CMCSharedToken

CMCSharedToken 工具为共享保障的 CMC 请求加密用户密码短语。例如：

```
$ CMCSharedToken -d . -p myNSSPassword -s "shared_passphrase" -o cmcSharedTok2.b64 -n
"subsystemCert cert-pki-tomcat"
```

共享密码短语(**-s**)被加密并存储在 **cmcSharedtok2.b64** 文件中(**-o**) 使用名为 **subsystemCert-pki-tomcat** (**-n**) 在当前目录中的 NSS 数据库中的证书。**默认安全令牌内部使用（因为未指定 -h ），并且 myNSSPassword 的令牌密码用于访问令牌。**

详情请查看 **CMCSharedtoken(1) man page** 和 第 7.2.1 节 "使用 **CMCRequest**撤销证书"。

2.5.8. CRMFPopClient

CRMFPopClient 实用程序是使用 **NSS** 数据库的证书请求消息格式(**CRMF**)客户端，并提供 **Possession** 的概念验证。

例如：

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b kra.transport -w
"AES/CBC/PKCS5Padding" -t false -v -o /user_or_entity_database_directory/example.csr
```

此示例使用 **cn=subject_name subject DN(-n)**、在当前目录中的 **NSS** 数据库创建一个新的 **CSR**，用于传输 **kra.transport (-b)**、**AES/CBC/PKCS5adding key wrap algorithm from the key wrap algorithm is specified**（ **-v**）和生成的 **CSR** 写入 **/user_or_entity_database_directory/example.csr** 文件(**-o**)。

有关详情、更多选项和其他示例，请参阅 CRMFPopClient --help 命令的输出以及 第 7.2.1 节 “使用 CMCRequest撤销证书”。

2.5.9. HttpClient

HttpClient 程序是一个 NSS 感知 HTTP 客户端，用于提交 CMC 请求。

例如：

```
$ HttpClient request.cfg
```

注意

HttpClient 实用程序的所有参数都存储在 request.cfg 文件中。如需更多信息，请参阅 HttpClient --help 命令的输出。

2.5.10. OCSPClient

一个在线证书状态协议(OCSP)客户端，用于检查证书撤销状态。

例如：

```
$ OCSPClient -h server.example.com -p 8080 -d /etc/pki/pki-tomcat/alias -c "caSigningCert cert-pki-ca" --serial 2
```

这个示例在端口 8080 上查询 server.example.com OCSP 服务器(-h)检查由 caSigningcet cert-pki-ca (-c)签名的证书是否有效。 使用 /etc/pki/pki-tomcat/alias 目录中的 NSS 数据库。

如需了解更多详细信息、更多选项和其他示例，请参阅 OCSPClient --help 命令的输出。

2.5.11. PKCS10Client

PKCS10Client 工具为 RSA 和 EC 密钥（可选）在 HSM 中创建一个 PKCS10 格式的 CSR。

例如：

```
$ PKCS10Client -d /etc/dirsrv/slapd-instance_name/ -p password -a rsa -l 2048 -o ~/ds.csr -n
"CN=$HOSTNAME"
```

这个示例在 /etc/dirsrv/slapd-instance_name/ 目录中（-d with database password (-p)）中创建带有 2048 字节(-l)密钥。输出 CSR 存储在 ~/ds.cfg 文件中(-o)，证书 DN 为 CN=$HOSTNAME (-n)。

详情请查看 PKCS10Client(1) man page。

2.5.12. PrettyPrintCert

PrettyPrintCert 工具以人类可读格式显示证书的内容。

例如：

```
$ PrettyPrintCert ascii_data.cert
```

此命令解析 ascii_data.cert 文件的输出，并以人类可读格式显示其内容。输出中包含签名算法、exponent、modulus 和证书扩展等信息。

详情请查看 PrettyPrintCert(1) man page。

2.5.13. PrettyPrintCrl

PrettyPrintCrl 实用程序以人类可读格式显示 CRL 文件的内容。

例如：

```
$ PrettyPrintCrl ascii_data.crl
```

此命令解析 ascii_data.crl 的输出，并以人类可读格式显示其内容。输出中包括信息，如撤销签名算法、撤销者以及已撤销证书及其原因列表。

详情请查看 PrettyPrintCrl(1) man page。

2.5.14. TokenInfo

TokenInfo 实用程序列出了 NSS 数据库中的所有令牌。

例如：

```
$ TokenInfo ./nssdb/
```

此命令列出在指定数据库目录中注册的所有令牌（HSM、软令牌等）。

如需了解更多详细信息、更多选项和其他示例，请参阅 TokenInfo 命令的输出

2.5.15. tkstool

tkstool 程序与令牌密钥服务(TKS)子系统交互。

例如：

```
$ tkstool -M -n new_master -d /var/lib/pki/pki-tomcat/alias -h token_name
```

该命令在 HSM token_name上的 /var/lib/pki/pki-tomcat/alias NSS 数据库中创建一个名为 new_master (-n)的新 master 密钥(-M)

详情请查看 tkstool -H 命令的输出。

2.6. ENTERPRISE 安全客户端

Enterprise Security Client 是 Red Hat Certificate Systemnbsp 的工具；授权证书 Red Hat Certificate Systemnbsp;System 简化了智能卡管理。最终用户可以使用安全令牌（smart 卡）来存储用于应用程序的用户证书，如单点登录访问和客户端身份验证。最终用户会签发包含签名、加密和其他加密功能所需的证书和密钥的令牌。

Enterprise Security Client 是 CertificateCertificate Systemnbsp 的第三个部分；系统的完整令牌管理系统。两个子系统 - 令牌密钥服务(TKS)和令牌处理系统(TPS)用于处理与令牌相关的操作。Enterprise Security Client 是允许智能卡和用户访问令牌管理系统的接口。

注册令牌后，可将 Mozilla Firefox 和 Thunderbird 等应用程序配置为识别令牌并将其用于安全操作，如客户端身份验证和 S/MIME 邮件。Enterprise Security Client 提供以下功能：

- 支持 JavaCard 2.1 或更高版本卡和全球平台 2.01- 兼容智能卡，如 Safenet 的 330J 智能卡。
- 支持 Gemalto TOP IM FIPS CY2 令牌，包括智能卡和 GemPCKey USB 形式因素键。
- 支持 SafeNet 智能卡 650(SC650)。
- 注册安全令牌，以便被 TPS 识别。
- 维护安全令牌，如使用 TPS 重新注册令牌。
- 提供有关受管理的令牌或令牌的当前状态的信息。
- 支持服务器端密钥生成，以便在令牌丢失时可以在单独的令牌中存档并恢复密钥。

Enterprise Security Client 是最终用户在智能卡或令牌上注册和管理密钥和证书的客户端。这是 CertificateCertificate Systemnbsp;System 令牌管理系统(TPS)和令牌密钥服务(TKS)中的最后一个组件。

Enterprise Security Client 提供令牌管理系统的用户界面。最终用户可以签发安全令牌，其中包含签名、加密和其他加密功能所需的证书和密钥。要使用令牌，TPS 必须能够识别并与之通信。企业安全客户端是要注册令牌的方法。

企业安全客户端通过 SSL/TLS HTTP 频道与 TPS 的后端通信。它基于用户界面的可扩展 Mozilla XULRunner 框架，同时为基于简单的 HTML 的 UI 保留传统的 Web 浏览器容器。

在令牌被正确注册后，可将 Web 浏览器配置为识别令牌并将其用于安全操作。Enterprise Security Client 提供以下功能：

- 允许用户注册安全令牌，以便被 TPS 识别。
- 允许用户维护安全令牌。例如，企业安全客户端可以使用 TPS 重新注册令牌。
- 通过默认的和自定义令牌配置集，为多种不同类型的令牌提供支持。默认情况下，TPS 可以自动注册用户密钥、设备密钥和安全分析密钥。可以添加额外的配置集，以便使用不同的使用的令牌（由令牌 CUID 等属性标识）根据适当的配置集自动注册。
- 提供有关被管理的令牌的当前状态的信息。

部分 II. 设置证书服务

第 3 章 制作发行证书规则（证书配置文件）

CertificateCertificate Systemnbsp;System 提供了一个自定义的框架，用于应用传入请求的策略，并控制输入请求类型和输出证书类型，这些框架称为 证书配置文件。证书配置集在证书管理器端点页面中设置证书注册表单所需的信息。本章论述了如何配置证书配置集。

3.1. 关于证书配置集

证书配置集定义了与发布特定类型的证书相关联的一切内容，包括身份验证方法、授权方法、默认证书内容、内容值的限制以及证书配置文件的输入和输出内容。注册和续订请求将提交到证书配置文件，然后受该证书配置集中设置的默认值和约束。这些限制通过与证书配置集关联的输入表单或通过其它方法提交请求。从证书配置文件请求发布的证书包含默认设置所需内容，以及默认参数所需的信息。约束提供了证书中允许的内容的规则。

有关使用和自定义证书配置集的详情，请参考 第 3.2 节 “设置证书配置集”。

证书系统包含一组默认配置集。虽然创建了默认配置集来满足大多数部署，但每个部署都可以添加自己的新证书配置集或修改现有的配置集。

- 身份验证.在每个认证配置集中都可以指定身份验证方法。
- 授权.在每个认证配置集中都可以指定授权方法。
- 配置集输入。配置集输入是请求证书时提交给 CA 的参数和值。配置集输入包括证书请求的公钥和证书实体请求的证书主题名称。
- 配置集输出。配置集输出是参数和值，用于指定向端点提供证书的格式。当请求成功时，配置集输出为 CMC 响应，其中包含 PKCS#7 证书链。
- 证书内容。每个证书都定义内容信息，如为其分配的实体的名称（主题名称）、签名算法及其有效周期。在 X.509 标准中定义证书中包含的内容。使用 X509 标准的版本 3，证书也可以包含扩展。有关证书扩展的详情请参考 第 B.3 节 “标准 X.509 v3 证书扩展参考”。

关于证书配置文件的所有信息都在配置集的配置文件中的 set 条目中定义。当应该同时请求多个证书时，可以在配置集策略中定义多个集合条目来满足每个证书的需求。每个策略集都包含

多个策略规则，每个策略规则描述了证书内容中的一个字段。策略规则可以包括以下部分：

- 配置集默认值。以下是证书中包含的信息的预定义参数和允许值。配置集默认值包括证书的有效性周期，以及每个发布的证书扩展显示哪些证书扩展。

- 配置集限制。约束设置规则或策略以发布证书。另外，配置集限制包括需要证书主体名称至少包含一个 CN 组件的规则，从而将证书的有效性设置为 360 天，以定义续订允许的宽限期，或者要求 subjectaltname 扩展始终设置为 true。

3.1.1. 注册配置集

定义输入、输出和策略集合的每个配置集的参数在表 11.1 中被详细列出。红帽认证系统规划、安装和部署指南中的配置文件参数.

配置集通常包含输入、策略集和输出，如 例 3.1 “caCMCUserCert Profile 示例” 中的 caUserCert 配置集所示。

> **例 3.1. caCMCUserCert Profile 示例**
>
> 证书配置集的第一部分是描述。这会显示名称、长描述（无论是否启用）以及谁启用它。
>
> ```
> desc=This certificate profile is for enrolling user certificates by using the CMC certificate
> request with CMC Signature authentication.
> visible=true
> enable=true
> enableBy=admin
> name=Signed CMC-Authenticated User Certificate Enrollment
> ```
>
> 
>
> **注意**
>
> 这个配置集中缺少 auth.instance_id= 条目意味着，不需要使用此配置集进行身份验证来提交注册请求。但是，经授权 CA 代理的手动批准需要经过授权后才能获得帮助。

接下来，配置集会列出配置集的所有所需输入：

```
input.list=i1
input.i1.class_id=cmcCertReqInputImp
```

对于 caCMCUserCert 配置集，这定义了证书请求类型，即 CMC。

接下来，配置集必须定义输出，即最终证书的格式。唯一可用的是 certOutputImpl，这会导致 CMC 响应在成功时返回到请求者。

```
output.list=o1
output.o1.class_id=certOutputImpl
```

最后最大的 - 配置块是配置集设置的策略。策略集列出了应用于最终证书的所有设置，如其有效期期、其续订设置以及证书可用于的操作。policyset.list 参数标识应用于某个证书的策略的块名称，policyset.userCertSet.list 列出了要应用的单个策略。

例如，根据策略中的配置，第六个策略会自动填充证书中的键使用范围。它通过设置限制来设置默认值，并要求证书使用这些默认值：

```
policyset.list=userCertSet
policyset.userCertSet.list=1,10,2,3,4,5,6,7,8,9
...
policyset.userCertSet.6.constraint.class_id=keyUsageExtConstraintImpl
policyset.userCertSet.6.constraint.name=Key Usage Extension Constraint
policyset.userCertSet.6.constraint.params.keyUsageCritical=true
policyset.userCertSet.6.constraint.params.keyUsageDigitalSignature=true
policyset.userCertSet.6.constraint.params.keyUsageNonRepudiation=true
policyset.userCertSet.6.constraint.params.keyUsageDataEncipherment=false
policyset.userCertSet.6.constraint.params.keyUsageKeyEncipherment=true
policyset.userCertSet.6.constraint.params.keyUsageKeyAgreement=false
policyset.userCertSet.6.constraint.params.keyUsageKeyCertSign=false
policyset.userCertSet.6.constraint.params.keyUsageCrlSign=false
policyset.userCertSet.6.constraint.params.keyUsageEncipherOnly=false
policyset.userCertSet.6.constraint.params.keyUsageDecipherOnly=false
policyset.userCertSet.6.default.class_id=keyUsageExtDefaultImpl
policyset.userCertSet.6.default.name=Key Usage Default
policyset.userCertSet.6.default.params.keyUsageCritical=true
policyset.userCertSet.6.default.params.keyUsageDigitalSignature=true
policyset.userCertSet.6.default.params.keyUsageNonRepudiation=true
policyset.userCertSet.6.default.params.keyUsageDataEncipherment=false
policyset.userCertSet.6.default.params.keyUsageKeyEncipherment=true
policyset.userCertSet.6.default.params.keyUsageKeyAgreement=false
policyset.userCertSet.6.default.params.keyUsageKeyCertSign=false
policyset.userCertSet.6.default.params.keyUsageCrlSign=false
```

```
policyset.userCertSet.6.default.params.keyUsageEncipherOnly=false
policyset.userCertSet.6.default.params.keyUsageDecipherOnly=false
...
```

3.1.2. 证书扩展：默认和限制

扩展配置用于包含在证书或规则中有关如何使用证书的附加信息。这些扩展可以在证书请求中指定，或者从配置集默认定义中获取，然后由限制强制实施。

如果在配置集中添加或标识了证书扩展，方法是添加与扩展名对应的默认值，并在请求中未设置证书扩展。例如： **Basic Constraints Extension** 标识证书是否为 **CA** 签名证书、可在 **CA** 中配置的最大下级 **CA** 数以及扩展是否重要（必需）：

```
policyset.caCertSet.5.default.name=Basic Constraints Extension Default
policyset.caCertSet.5.default.params.basicConstraintsCritical=true
policyset.caCertSet.5.default.params.basicConstraintsIsCA=true
policyset.caCertSet.5.default.params.basicConstraintsPathLen=-1
```

扩展也可以为名为 **constraints** 的证书请求设置所需的值。如果请求的内容与集合约束不匹配，则请求将被拒绝。约束通常对应于扩展名默认值，但并不总是对应。例如：

```
policyset.caCertSet.5.constraint.class_id=basicConstraintsExtConstraintImpl
policyset.caCertSet.5.constraint.name=Basic Constraint Extension Constraint
policyset.caCertSet.5.constraint.params.basicConstraintsCritical=true
policyset.caCertSet.5.constraint.params.basicConstraintsIsCA=true
policyset.caCertSet.5.constraint.params.basicConstraintsMinPathLen=-1
policyset.caCertSet.5.constraint.params.basicConstraintsMaxPathLen=-1
```

注意

要允许用户在证书请求中嵌入用户提供的扩展并忽略配置集中定义的默认值，配置集需要包含 **User Supplied Extension Default**，如 第 B.1.32 节 “用户补充默认设置”。

3.1.3. 输入和输出

输入设置信息，必须提交以接收证书。这可以是请求者信息、证书请求特定格式或机构信息。

配置集中配置的输出定义签发的证书格式。

在 CertificateCertificate Systemnbsp;System 中，通过 注册表单 （通过结束日期页面访问）访问配置文件。（即使客户端，如 TPS，通过这些表单提交注册请求。） 然后，输入对应于注册表单中的字段。输出与证书检索页面中包含的信息对应。

3.2. 设置证书配置集

在证书系统中，您可以添加、删除和修改注册的配置集：

- 使用 PKI 命令行界面
- 使用基于 Java 的管理控制台

本节提供有关每种方法的信息。

3.2.1. 使用 PKI 命令行界面管理证书注册配置集

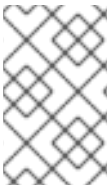
这部分论述了如何使用 pki 工具管理证书配置集。详情请查看 pki-ca-profile(1) man page。

注意

建议使用 raw 格式。有关配置集的每个属性和字段的详情，请参阅红帽认证系统规划、安装和部署指南中的有关文件系统创建和编辑证书配置集部分。

3.2.1.1. 启用和禁用证书配置集

在编辑证书配置集前，您必须禁用它。修改完成后，您可以重新启用该配置集。



注意

只有 CA 代理可以启用和禁用证书配置集。

例如，禁用 caCMCECserverCert 证书配置集：

```
# pki -c password -n caagent ca-profile-disable caCMCECserverCert
```

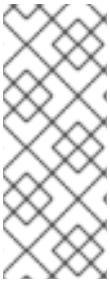
例如，启用 caCMCECserverCert 证书配置集：

```
# pki -c password -n caagent ca-profile-enable caCMCECserverCert
```

3.2.1.2. 以 Raw 格式创建证书配置集

要以 raw 格式创建新配置集：

```
# pki -c password -n caadmin ca-profile-add profile_name.cfg --raw
```



注意

在 raw 格式中，指定新的配置集 ID，如下所示：

```
profileId=profile_name
```

3.2.1.3. 使用 Raw 格式编辑证书配置文件

CA 管理员可以使用 raw 格式编辑证书配置文件，而无需手动下载配置文件。

例如，要编辑 caCMCECserverCert 配置集：

```
# pki -c password -n caadmin ca-profile-edit caCMCECserverCert
```

此命令自动下载原始格式的配置集配置，并在 VI 编辑器中打开它。关闭编辑器时，服务器上的配置集配置会更新。

编辑配置集后您不需要重启 CA。

重要

在编辑配置集前，禁用配置集。详情请查看 第 3.2.1.1 节 “启用和禁用证书配置集”。

例 3.2. 以 RAW 格式编辑证书配置集

例如，要编辑 caCMCserverCert 配置集以接受多个用户提供的扩展：

1. 禁用配置集作为 CA 代理：

```
# pki -c password -n caagemt ca-profile-disable caCMCserverCert
```

2. 以 CA 管理员身份编辑配置集：

   a. 在 VI 编辑器中下载并打开配置集：

   ```
   # pki -c password -n caadmin ca-profile-edit caCMCserverCert
   ```

   b. 更新配置以接受扩展。详情请查看 *例 B.3 “CSR 中的多个用户提供的扩展”*。

3. 将配置集启用为 CA 代理：

```
# pki -c password -n caagent ca-profile-enable caCMCserverCert
```

3.2.1.4. 删除证书配置集

删除证书配置集：

```
# pki -c password -n caadmin ca-profile-del profile_name
```

重要

在删除配置集前，禁用配置集。详情请查看 *第 3.2.1.1 节 “启用和禁用证书配置集”*。

3.2.2. 使用基于 Java 的管理控制台管理证书注册配置集

3.2.2.1. 通过 CA 控制台创建证书配置集

为了安全起见，证书系统可强制隔离现有的证书配置集，只有管理员在代理允许后才可以编辑它。要添加新证书配置集或修改现有证书配置集，请以管理员身份执行以下步骤：

1. 登录到证书Certificate Systemnbsp;System CA 子系统控制台。

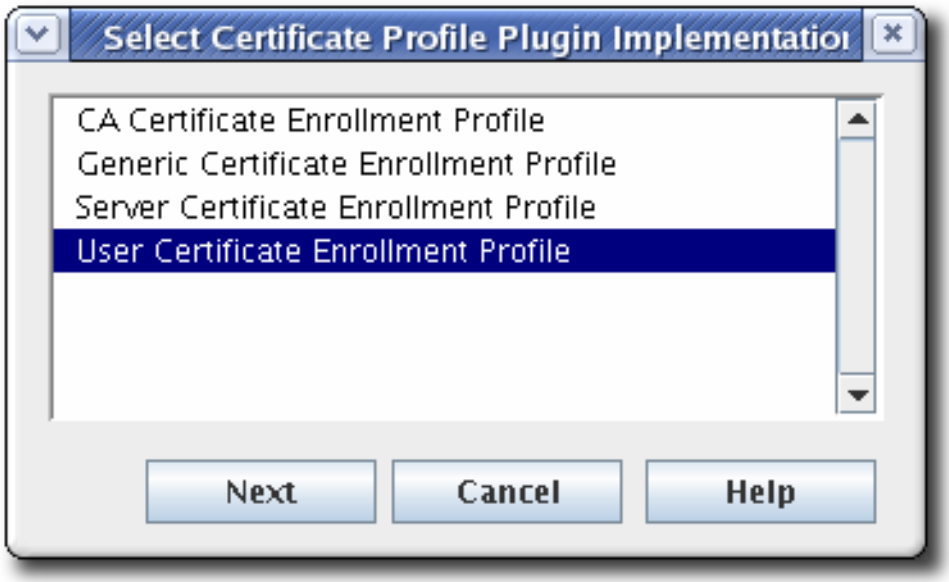
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，选择 Certificate Manager，然后选择 Certificate Profiles。

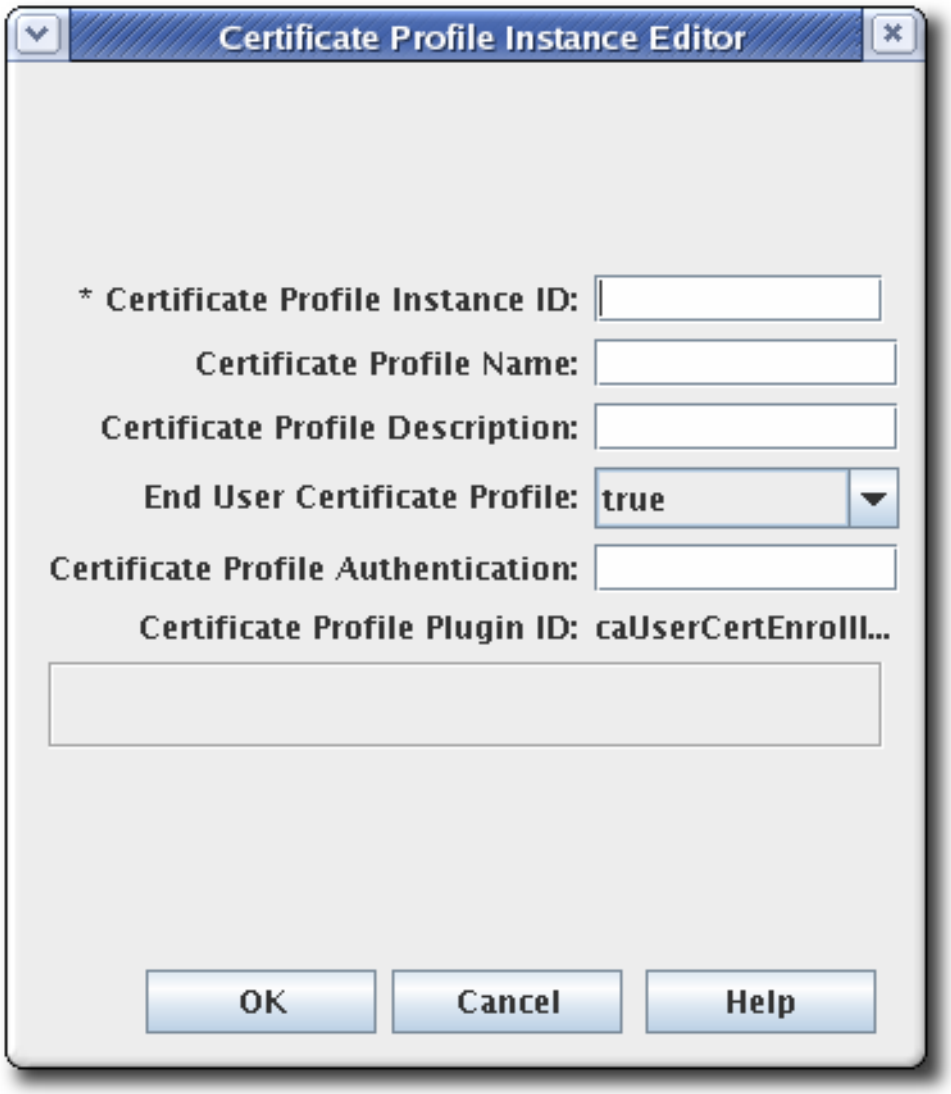
   Certificate Profile Instances Management 选项卡（列出配置的证书配置集）会打开。

3. 要创建新证书配置文件，请单击 Add。

   在 Select Certificate Profile Plugin Implementation 窗口中，选择创建配置集的证书类型。



4. 在 Certificate Profile Instance Editor 中查看配置集信息。



- *证书配置文件实例 ID.这是系统用来识别配置集的 ID。*
- *证书配置文件名称.这是配置集的用户友好名称。*
- *证书配置文件描述.*
- *最终用户证书配置文件.这将设置请求是否必须通过配置集的输入表单进行发布。这通常设置为 true。将其设置为 false 可让通过证书管理器的证书配置集框架处理签名请求，而不是通过证书配置集的输入页面。*
- *证书配置文件身份验证.这将设置身份验证方法。通过为身份验证提供实例 ID 来设置自动化身份验证。如果此字段为空，则验证方法是代理批准的注册；请求将提交到代理服务接口的请求队列。*

除非用于 TMS 子系统，否则管理员必须选择以下身份验证插件之一：

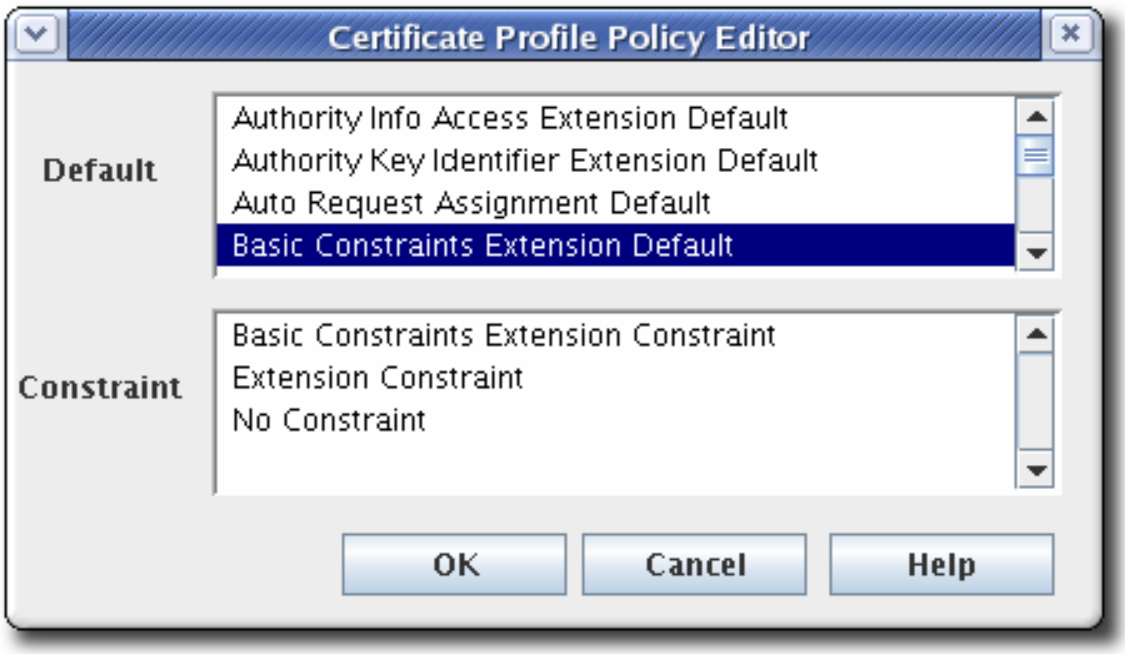
- CMCAuth ：当 CA 代理必须批准并提交注册请求时，使用此插件。
- CMCUserSignedAuth ：使用此插件使非代理用户注册自己的证书。

5. 点击 OK。插件编辑器关闭，新配置集在 profile 选项卡中列出。

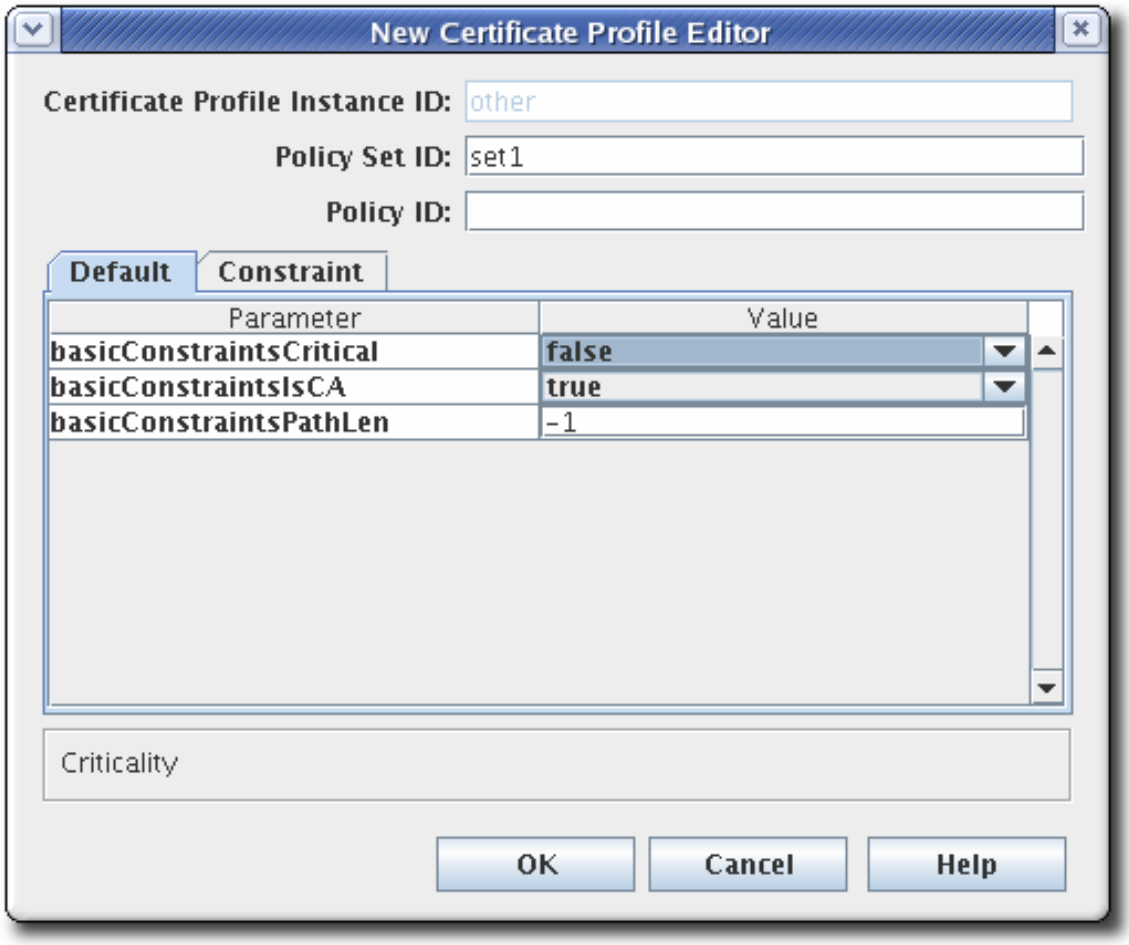
6. 为新配置集配置策略、输入和输出。从列表中选择新配置集，再单击 Edit/View。

7. 在 证书配置文件规则编辑器窗口的 Policies 选项卡中设置策略。Policies 标签页列出了为配置集类型默认设置的策略。

   a. 要添加策略，请点击 Add。



   b. 从 Default 字段中选择默认值，在 Constraints 字段中选择与该策略关联的限制，然后单击确定。

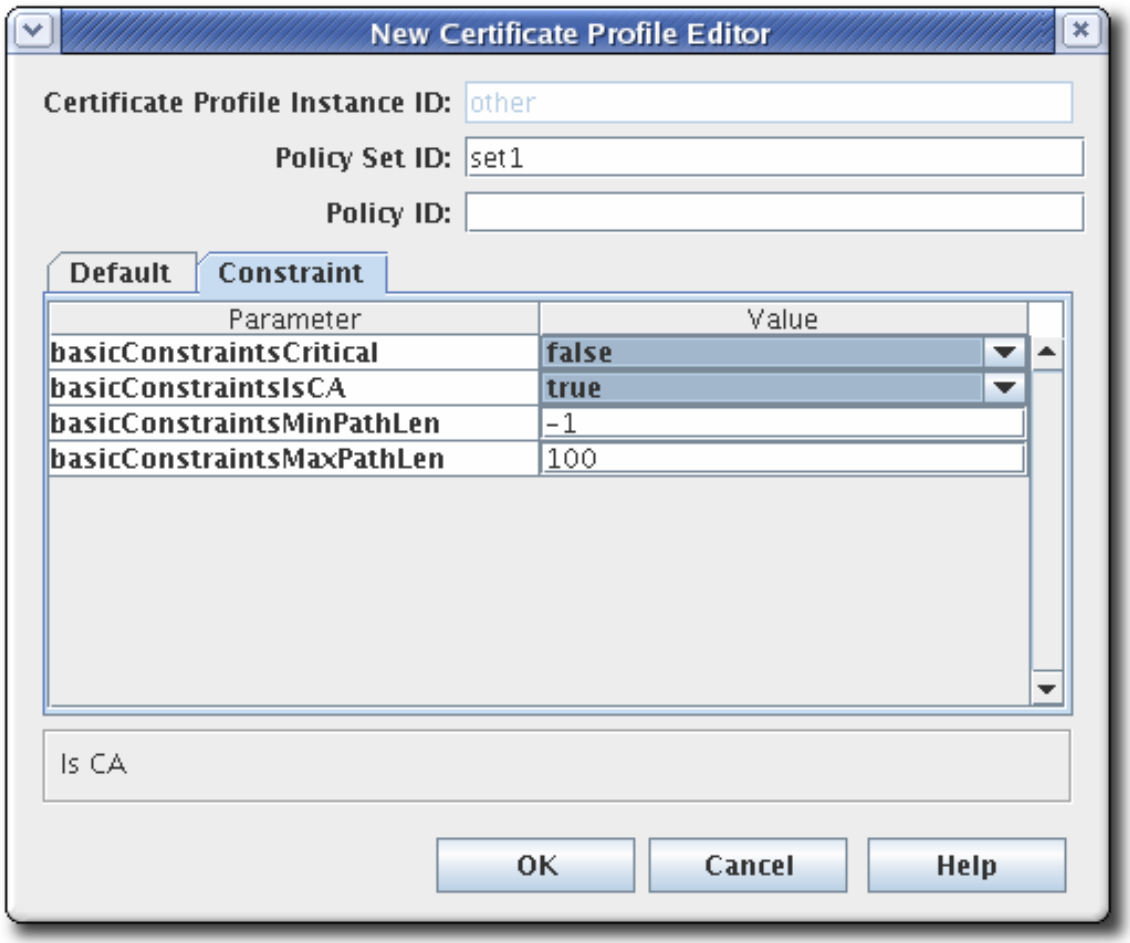


c.

填写策略设置 ID。在发出双键对时，单独的策略会定义与每个证书关联的策略。然后填写证书配置集策略 ID，这是证书配置集策略的名称或标识符。

d.

在 Defaults 和 Constraints 选项卡中配置任何参数。



默认值 定义填充证书请求的属性，后者决定了证书的内容。这些可以是扩展、有效期期或证书中包含的其他字段。约束 定义了默认值。

如需每个默认或约束的完整详情，请参阅 第 B.1 节 “默认参考” 和 第 B.2 节 “约束参考”。

要修改现有策略，请选择一个策略，然后点 Edit。然后，编辑该策略的默认值和限制。

要删除策略，请选择策略，然后点 Delete。

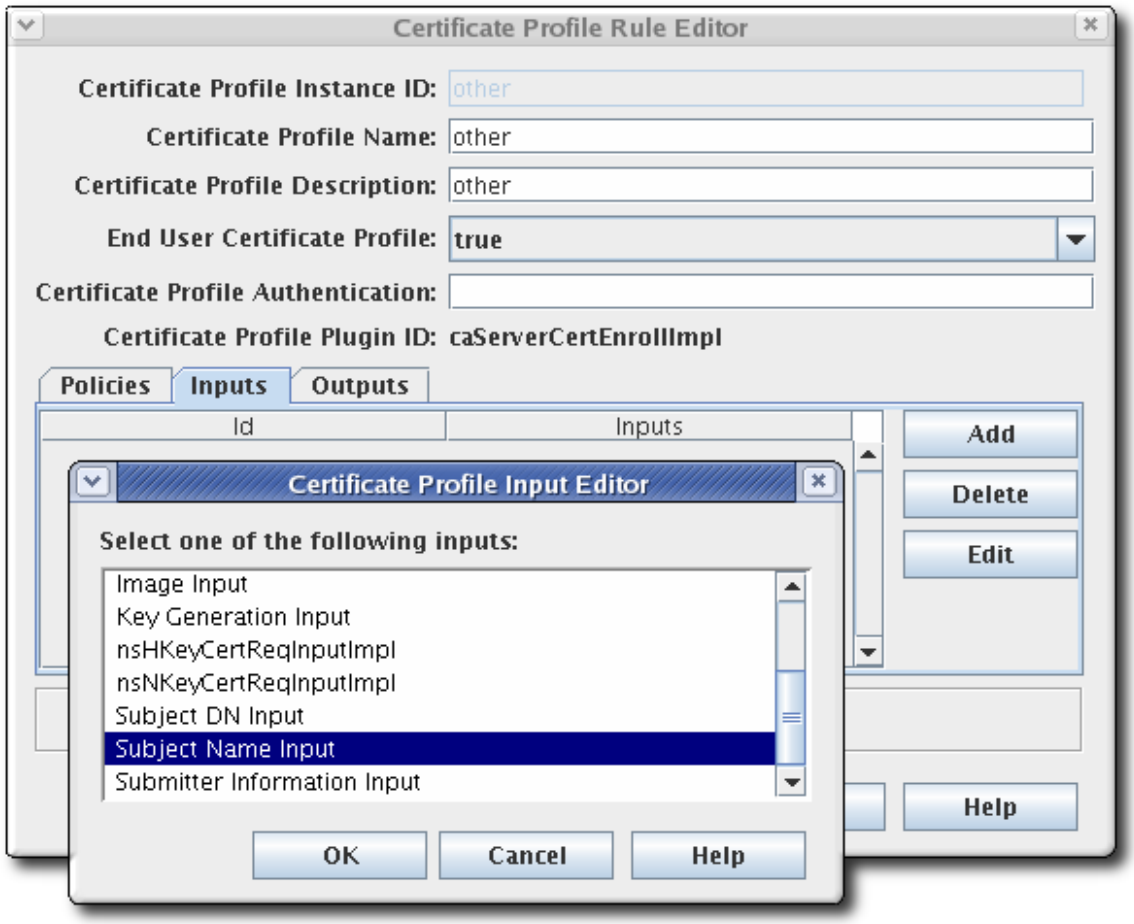
8. *在 证书配置文件规则编辑器窗口的 输入 选项卡中设置输入。配置集可以有多个输入类型。*

注意

除非为 TMS 子系统配置配置集，否则仅选择 cmcCertReqInput 并删除其他配置集，并点 Delete 按钮。

a.

*要添加输入，请单击 **Add**。*

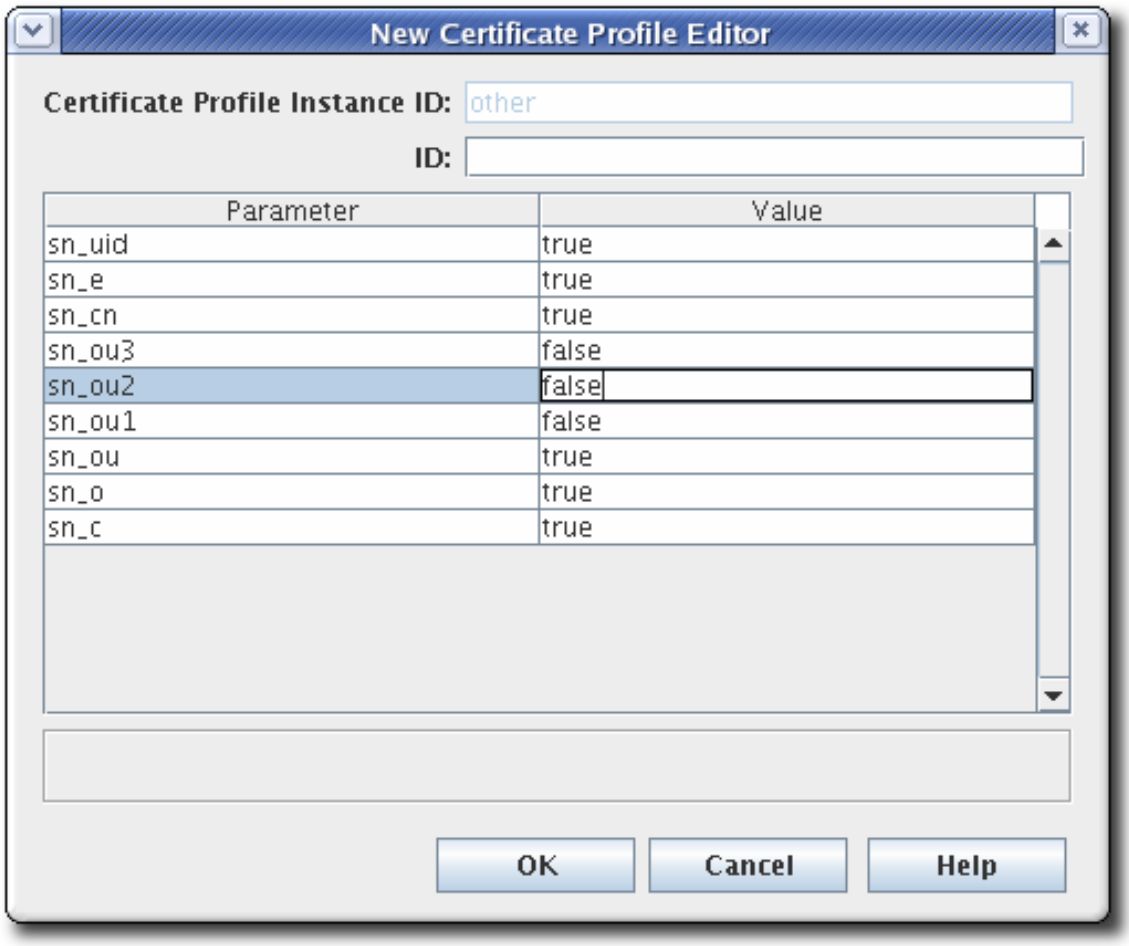


b.

从列表中选择输入，然后单击确定。如需默认输入的完整详情，请参阅 第 A.1 节 “输入参考”。

c.

*此时会打开 **New Certificate Profile Editor** 窗口。设置输入 **ID**，然后单击确定。*



可以添加和删除输入。可以选择编辑输入，但因为输入没有参数或其他设置，因此没有配置任何内容。

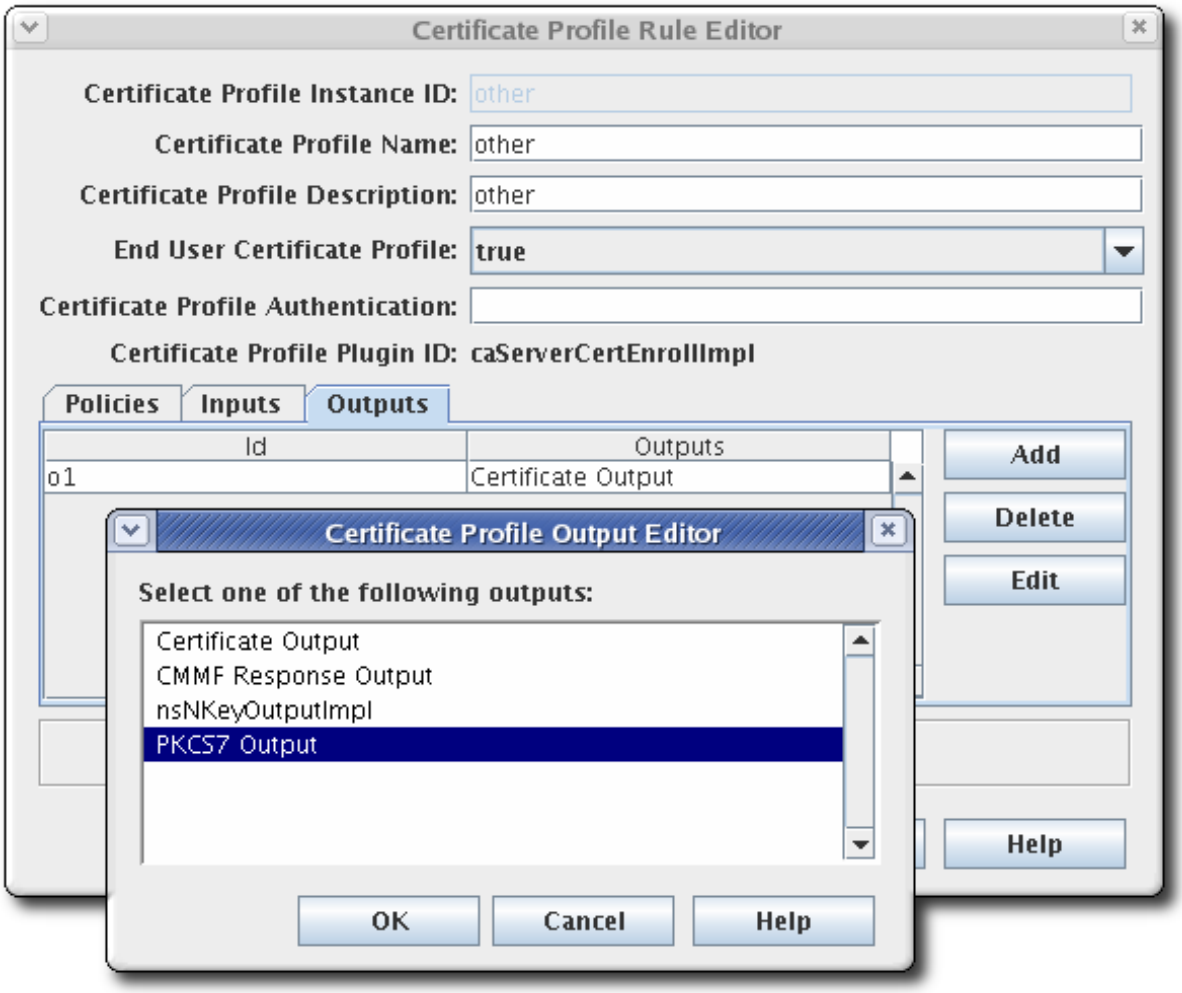
*要删除输入，请选择输入，然后单击 **Delete**。*

9.

*在 证书配置集 规则编辑器 窗口的 **Outputs** 选项卡中设置输出。*

必须为使用自动身份验证方法的任何证书配置集设置输出；不需要为使用代理批准的验证的任何证书配置集设置输出。默认情况下，所有配置集都会设置证书输出类型，并自动添加到自定义配置集。

*除非为 **TMS** 子系统配置配置集，否则仅选择 **certOutput**。*



可以添加和删除输出。可以选择编辑输出，但由于输出没有参数或其他设置，所以没有配置任何设置。

a. *要添加输出，请单击* **Add**。

b. *选择列表中的输出，然后单击确定。*

c. *为输出指定名称或标识符，然后单击* **OK**。

此输出将在输出标签页中列出。您可以编辑它，向此输出中的参数提供值。

要删除输出，请选择列表中的输出，然后单击 **Delete**。

10. 重启 CA 以应用新配置集。

```
systemctl restart pki-tomcatd-nuxwdog@instance_name.service
```

11. 以管理员创建配置集后，CA 代理必须在代理服务页面中批准配置集以启用该配置集。

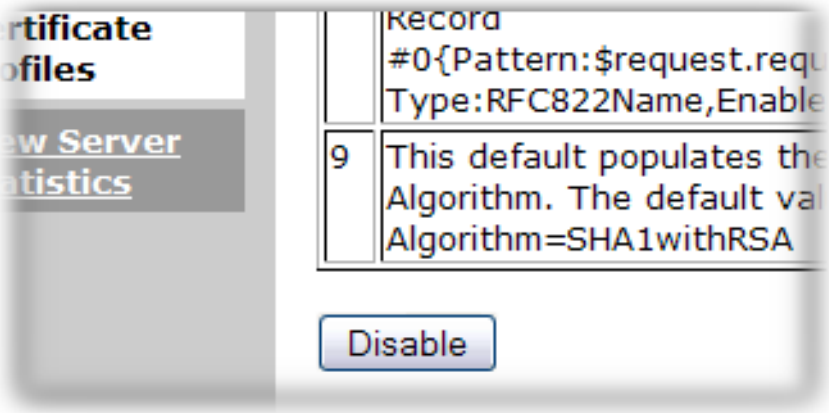
a. 打开 CA 的服务页面。

```
https://server.example.com:8443/ca/services
```

b. 单击 **Manage Certificate Profiles** 链接。本页列出了管理员设置的所有证书配置文件，包括活跃和不活跃状态。

c. 点要批准的证书配置集的名称。

d. 在页面底部，单击 启用 按钮。



注意

如果这个配置集将与 TPS 搭配使用，则必须将 TPS 配置为识别配置集类型。这在 11.1.4 中。在红帽认证系统规划、安装和部署指南中管理智能卡 CA 配置文件.

配置集的授权方法只能使用命令行添加到配置集，如红帽证书系统规划、安装和部署指南中的文件系统中直接创建和编辑证书配置集部分。

3.2.2.2. 在控制台中编辑证书配置集

修改现有的证书配置集：

1. 登录到代理服务页面并禁用配置集。

   当一个证书配置集由代理启用后，证书配置集会在 Certificate Profile Instance Management 选项卡中被标记为证书配置集，且无法通过控制台以任何方式编辑证书配置集。

2. 登录到证书Certificate Systemnbsp;System CA 子系统控制台。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

3. 在 Configuration 选项卡中，选择 Certificate Manager，然后选择 Certificate Profiles。

4. 选择证书配置文件，再单击 Edit/View。

5. 此时会出现 证书 Profile Rule Editor 窗口。对默认值、约束、输入或输出的任何更改。

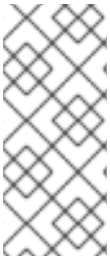
   注意

   配置集实例 ID 无法修改。

   如有必要，通过拉出窗口的某一角放扩大窗口。

6. 重启 CA 以应用更改。

7. 在 agent services 页面中，重新启用该配置集。



注意

删除代理不会批准的任何证书配置集。出现在 **Certificate Profile Instance Management** 选项卡中的任何证书配置集也会出现在代理服务界面中。如果已经启用了配置集，则代理必须禁用它，然后才能从配置集列表中删除。

3.2.3. 列出证书注册配置集

在安装 **CertificateCertificate Systemnbsp;System CA** 时，以下预定义的证书配置集可使用并在此环境中设置。这些证书配置集是为最常见的类型证书而设计的，它们提供常见的默认值、约束、身份验证方法、输入和输出。

要在命令行中列出可用配置集，请使用 **pki** 工具程序。例如：

```
# pki -c password -n caadmin ca-profile-find
------------------
59 entries matched
------------------
  Profile ID: caCMCserverCert
  Name: Server Certificate Enrollment using CMC
  Description: This certificate profile is for enrolling server certificates using CMC.

  Profile ID: caCMCECserverCert
  Name: Server Certificate wth ECC keys Enrollment using CMC
  Description: This certificate profile is for enrolling server certificates with ECC keys using CMC.

  Profile ID: caCMCECsubsystemCert
  Name: Subsystem Certificate Enrollment with ECC keys using CMC
  Description: This certificate profile is for enrolling subsystem certificates with ECC keys using CMC.

  Profile ID: caCMCsubsystemCert
  Name: Subsystem Certificate Enrollment using CMC
  Description: This certificate profile is for enrolling subsystem certificates using CMC.

  ...
----------------------------
Number of entries returned 20
```

详情请查看 **pki-ca-profile(1) man page**。如需更多信息，请参阅 *红帽认证系统规划、安装和部署指南*。

3.2.4. 显示证书注册配置集详情

例如，要显示特定的证书配置集，如 **caECFullCMCUserSignedCert** ：

```
$ pki -c password -n caadmin ca-profile-show caECFullCMCUserSignedCert
-----------------------------------
Profile "caECFullCMCUserSignedCert"
-----------------------------------
  Profile ID: caECFullCMCUserSignedCert
  Name: User-Signed CMC-Authenticated User Certificate Enrollment
  Description: This certificate profile is for enrolling user certificates with EC keys by using the CMC
certificate request with non-agent user CMC authentication.

  Name: Certificate Request Input
  Class: cmcCertReqInputImpl

    Attribute Name: cert_request
    Attribute Description: Certificate Request
    Attribute Syntax: cert_request

  Name: Certificate Output
  Class: certOutputImpl

    Attribute Name: pretty_cert
    Attribute Description: Certificate Pretty Print
    Attribute Syntax: pretty_print

    Attribute Name: b64_cert
    Attribute Description: Certificate Base-64 Encoded
    Attribute Syntax: pretty_print
```

例如，要显示特定的证书配置集，如 caECFullCMCUserSignedCert，采用 raw 格式：

```
$ pki -c password -n caadmin ca-profile-show caECFullCMCUserSignedCert --raw
#Wed Jul 25 14:41:35 PDT 2018
auth.instance_id=CMCUserSignedAuth
policyset.cmcUserCertSet.1.default.params.name=
policyset.cmcUserCertSet.4.default.class_id=authorityKeyIdentifierExtDefaultImpl
policyset.cmcUserCertSet.6.default.params.keyUsageKeyCertSign=false
policyset.cmcUserCertSet.10.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.10.constraint.name=Renewal Grace Period Constraint
output.o1.class_id=certOutputImpl

...
```

详情请查看 pki-ca-profile(1) man page。

3.3. 在配置集中定义密钥默认值

在创建证书配置集时，必须在 Subject Key Identifier Default 前添加 Key Default。CertificateCertificate Systemnbsp;System 在创建或应用 Subject Key Identifier Default 之前，在 Key Default 中处理密钥限制，因此如果密钥还没有被处理，在主题名称中设置密钥会失败。

例如，对象签名配置集可以定义这两个默认值：

```
policyset.set1.p3.constraint.class_id=noConstraintImpl
policyset.set1.p3.constraint.name=No Constraint
policyset.set1.p3.default.class_id=subjectKeyIdentifierExtDefaultImpl
policyset.set1.p3.default.name=Subject Key Identifier Default
...
policyset.set1.p11.constraint.class_id=keyConstraintImpl
policyset.set1.p11.constraint.name=Key Constraint
policyset.set1.p11.constraint.params.keyType=RSA
policyset.set1.p11.constraint.params.keyParameters=1024,2048,3072,4096
policyset.set1.p11.default.class_id=userKeyDefaultImpl
policyset.set1.p11.default.name=Key Default
```

在 policyset 列表中，必须在 Subject Key Identifier Default(p3)前列出 Key Default(p11)。

```
policyset.set1.list=p1,p2,p11,p3,p4,p5,p6,p7,p8,p9,p10
```

3.4. 配置配置集以启用续订

本节讨论如何为证书续订设置配置集。有关如何更新证书的详情请参考 第 5.5 节 “续订证书”。

允许续订的配置集通常由 renewGracePeriodConstraint 条目使用。例如：

```
policyset.cmcUserCertSet.10.constraint.class_id=renewGracePeriodConstraintImpl
policyset.cmcUserCertSet.10.constraint.name=Renewal Grace Period Constraint
policyset.cmcUserCertSet.10.constraint.params.renewal.graceBefore=30
policyset.cmcUserCertSet.10.constraint.params.renewal.graceAfter=30
policyset.cmcUserCertSet.10.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.10.default.name=No Default
```

3.4.1. 使用 Same Key 续订

允许为续订提交同一密钥的配置集，在 uniqueKeyConstraint 条目中将 allowSameKeyRenewal 参数设置为 true。例如：

```
policyset.cmcUserCertSet.9.constraint.class_id=uniqueKeyConstraintImpl
policyset.cmcUserCertSet.9.constraint.name=Unique Key Constraint
policyset.cmcUserCertSet.9.constraint.params.allowSameKeyRenewal=true
policyset.cmcUserCertSet.9.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.9.default.name=No Default
```

3.4.2. 使用新密钥续订

要使用新密钥续订证书，请使用与新密钥相同的配置集。证书系统使用用户签名证书的 subjectDN 签署新证书的请求。

3.5. 为证书设置签名算法

CA 的签名证书可以用 CA 支持的任何公钥算法为它的问题签名。例如，只要 CA 支持 ECC 和 RSA 算法，ECC 签名证书就可以为 ECC 和 RSA 证书请求签名。RSA 签名证书可以使用 EC 密钥为 PKCS #10 请求签名，但如果 ECC 模块不适用于 CA，则 CA 无法使用 EC 密钥为 PKCS #10 请求签名的 CRMF 证书请求，以验证 CRMF 概念验证(POP)。

ECC 和 RSA 是公钥加密和签名算法。两种公钥算法都支持不同的密码套件，用于加密和解密数据。CA 签名证书的功能部分是使用其支持的加密套件之一发布和签名的证书。

每个配置集可以定义 CA 应该用来为通过该配置集处理的证书签名的密码套件。如果没有设置签名算法，配置集将使用任何默认签名算法。

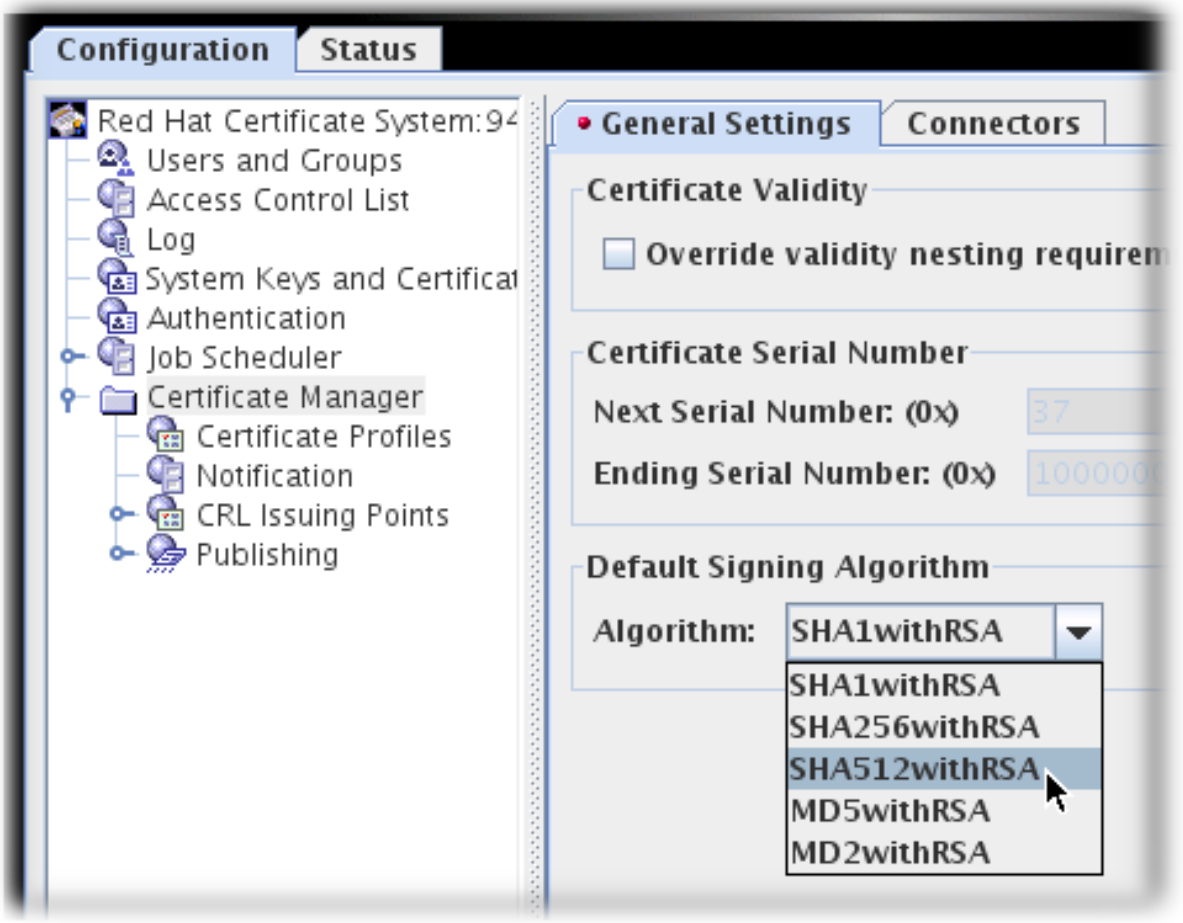
3.5.1. 设置 CA 的默认签名算法

1. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，展开 证书管理器树。

3. 在 General Settings 选项卡中，设置 Algorithm 下拉菜单中选择要使用的算法。



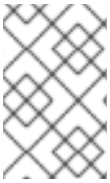
3.5.2. 在配置集中设置 Signing Algorithm Default

每个配置集都定义了一个 Signing Algorithm Default 扩展。默认有两个设置：如果证书请求指定了不同的算法，则默认算法和允许的算法列表。如果没有指定签名算法，则配置集会使用任何设置为 CA 的默认设置。

在配置集的 .cfg 文件中，算法设置了两个参数：

```
policyset.cmcUserCertSet.8.constraint.class_id=signingAlgConstraintImpl
policyset.cmcUserCertSet.8.constraint.name=No Constraint
policyset.cmcUserCertSet.8.constraint.params.signingAlgsAllowed=SHA256withRSA,SHA512
withRSA,SHA256withEC,SHA384withRSA,SHA384withEC,SHA512withEC
policyset.cmcUserCertSet.8.default.class_id=signingAlgDefaultImpl
policyset.cmcUserCertSet.8.default.name=Signing Alg
policyset.cmcUserCertSet.8.default.params.signingAlg=-
```

通过控制台配置 Signing Algorithm Default：



注意

在编辑配置集前，必须先由代理禁用它。

1. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

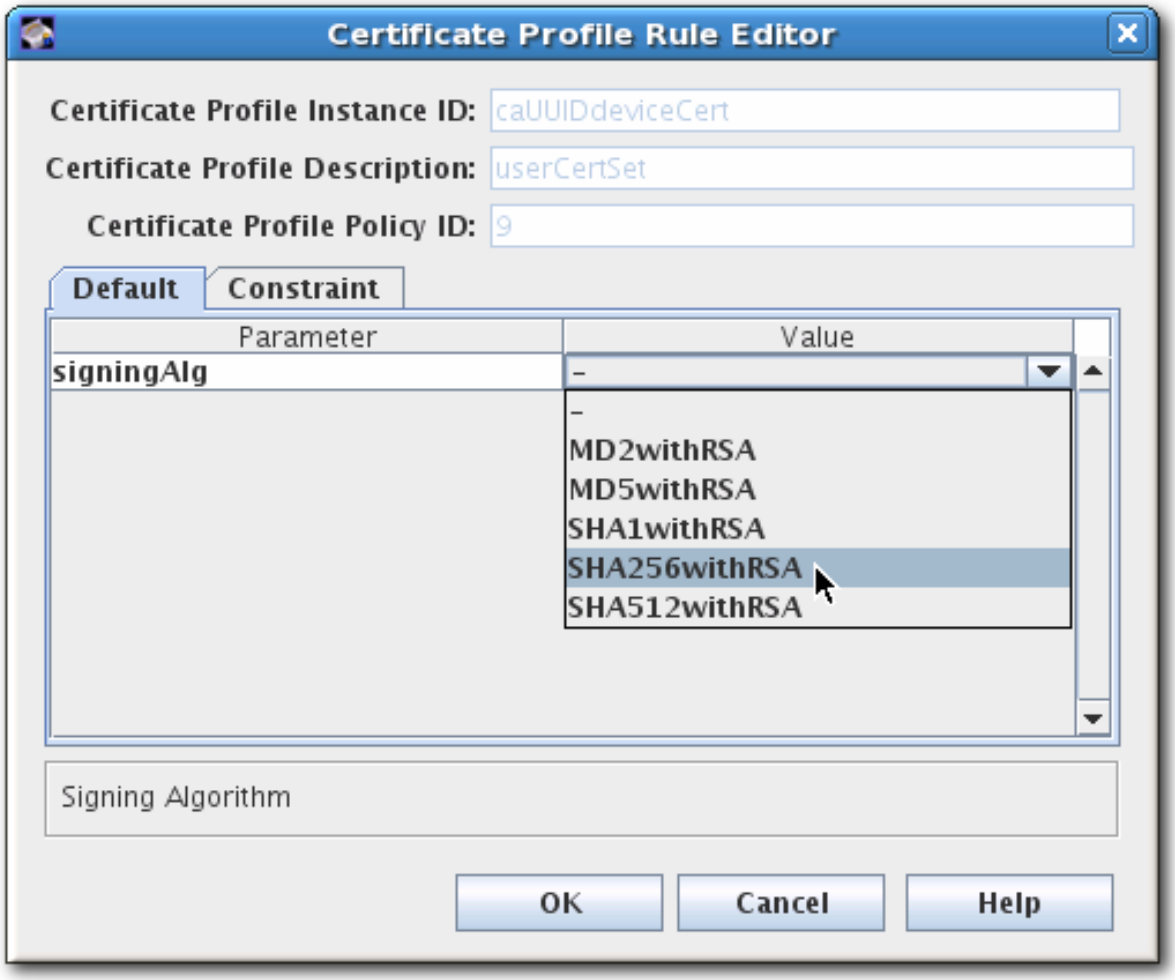
2. 在 Configuration 选项卡中，展开 证书管理器树。

3. 单击 Certificate Profiles 项。

4. 点 Policies 标签页。

5. 选择 Signing Alg 策略，然后单击 Edit 按钮。

6. 要设置默认的签名算法，请在 Defaults 选项卡中设置值。如果设为 -，则配置集将使用 CA 的默认。



7. *要设置一个可以在证书请求中接受的允许签名算法列表，打开 Constraints 标签页，然后在 Sign AlgsAllowed 的 Value 字段中设置算法列表。*

   约束的可能值列在 第 B.2.10 节 “签名算法” 中。

3.6. 管理 CA-RELATED 配置集

必须使用证书配置集和扩展来设置下级 CA 如何发出证书的规则。这里有两个部分：

- *管理 CA 签名证书*
- *定义可识别规则*

3.6.1. 在 CA 证书中设置限制

创建从属 CA 时，根 CA 可以对从属 CA 施加限制或限制。例如，根 CA 可以通过设置 CA 签名证书的 basicLenConstraint 字段来设置有效认证路径（允许的下级 CA 的数量）的最大深度信息。

证书链通常由实体证书、零个或者多个中间 CA 证书以及 root CA 证书组成。root CA 证书由外部可信 CA 自签名或签名。签发后，root CA 证书将作为可信 CA 加载到证书数据库中。

在发出 TLS 握手、发送 S/MIME 消息或发送签名对象时，证书交换会发生。作为握手的一部分，发送发送者应发送主题证书以及将主题证书链接到可信根所需的任何中间 CA 证书。要使证书链正常工作，证书应具有以下属性：

- CA 证书必须具有 Basic Constraints 扩展。
- CA 证书必须具有在 Key Usage 扩展中设置的 keyCertSign bit。
- 当 CA 生成新密钥时，它们必须将所有主题证书添加授权密钥标识符扩展。此扩展有助于将证书与旧的 CA 证书区分开来。CA 证书必须包含 Subject Key Identifier 扩展。

有关证书和扩展的更多信息，请参阅 互联网 X.509 公共密钥基础架构 - 证书和证书撤销列表 (CRL)Profile(RFC 5 280)。

这些扩展可以通过证书配置集注册页面配置。默认情况下，CA 包含所需的和合理的配置设置，但可以自定义这些设置。

注意

这个步骤描述了编辑 CA 证书配置集，该 CA 证书将 CA 证书发布到其下级 CA。

当配置 CA 实例是 /var/lib/pki/instance_name/ca/conf/caCert.profile 时使用的配置集。此配置集无法在 pkiconsole 中编辑（因为它仅在配置实例前可用）。在使用文本编辑器配置 CA 之前，可以在模板文件中编辑此配置集的策略。

修改 CA 使用的 CA 签名证书配置集中的默认设置：

1. 如果当前启用配置集，则必须在编辑配置集前禁用它。打开 agent services 页面，从左侧导

航菜单中选择 Manage Certificate Profiles，选择配置集，再单击 Disable profile。

2. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

3. 在 Configuration 选项卡的左侧导航树中，选择 Certificate Manager，然后选择 Certificate Profiles。

4. 从右窗口中选择 caCACert 或适当的 CA 签名证书配置集，然后点击 Edit/View。

5. 在 证书 配置文件规则编辑器的 Policies 选项卡中，选择并编辑 Key Usage 或 Extended Key Usage Extension Default（如果存在或将其添加到配置文件中）。

6. 选择 Key Usage 或 Extended Key Usage Extension 约束（根据情况考虑默认设置）。

7. 设置 CA 证书的默认值。如需更多信息，请参阅 第 B.1.13 节 “主要使用扩展默认值” 和 第 B.1.8 节 “扩展密钥使用扩展默认值”。

8. 设置 CA 证书的约束值。没有为键使用扩展设置限制；对于扩展密钥使用扩展，为 CA 设置适当的 OID 约束。更多信息请参阅 第 B.1.8 节 “扩展密钥使用扩展默认值”。

9. 对配置集进行了更改后，再次登录代理服务页面，然后重新启用证书配置文件。

有关修改证书配置集的详情请参考 第 3.2 节 “设置证书配置集”。

3.6.2. 在签发者证书上更改 CA 的限制

在配置子系统后，默认设置签发的证书的限制。包括：

- 是否可以发布证书有效期超过 CA 签名证书。默认值为不允许使用。

- 用于签名证书的签名算法。

- CA 可用于发布证书的序列号范围。

从属 CA 对有效期期、证书类型以及可以发布的扩展类型具有约束。从属 CA 可能会发布违反这些限制的证书，但客户端可以进行身份验证违反这些限制的证书。在更改下级 CA 的发布规则前，请检查 CA 签名证书上设置的限制。

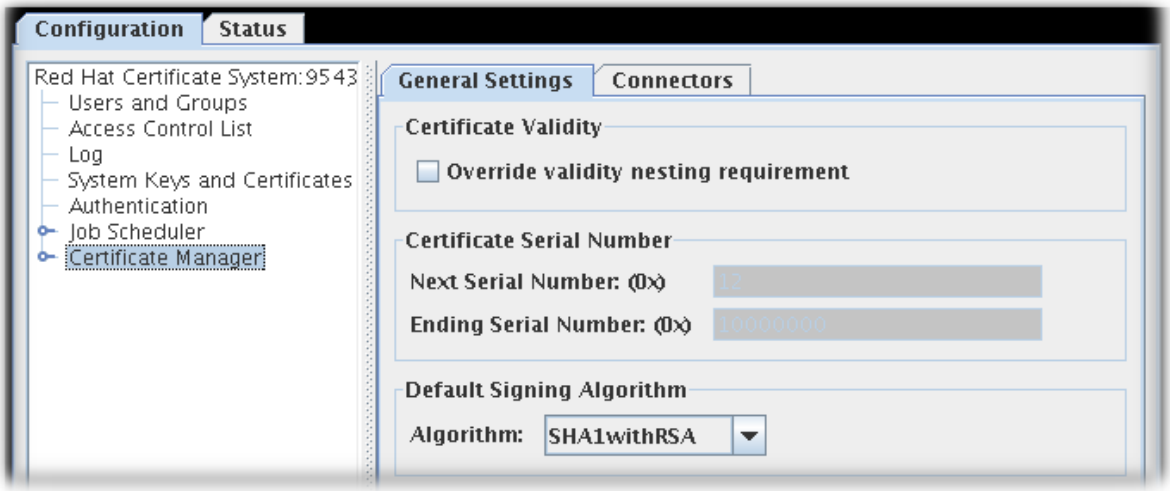
更改证书可保证规则：

1. 打开 CertificateCertificate Systemnbsp;System Console。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡左侧导航树中，选择 Certificate Manager 项。

图 3.1. 默认非子 CA 中的 General Settings 标签页



3. 默认情况下，在非克隆的 CA 中，证书管理器 菜单项中的 General Settings 选项卡包含以下选项：

- 覆盖有效嵌套要求。此复选框设置证书管理器是否可以超过 CA 签名证书的有效性期发布证书。

如果没有选择这个复选框，且 CA 收到一个有效期周期的时间超过 CA 签名证书的有效性期，它会在 CA 签名证书过期日自动截断有效周期。

- 证书串行号.这些字段显示证书管理器发布的证书的序列号范围。服务器为它发出的下一个证书以及 结束 序列号中的数字分配给它问题的最后证书，在 Next 序列号 中为其分配序列号。

序列号范围允许部署多个 CA，并平衡每个 CA 问题的证书数量。签发者名称和序列号的组合会唯一标识证书。

注意

带有克隆的 CA 的序列号范围是 fluid。所有克隆的 CAs 共享定义下一个可用范围的通用配置条目。当一个 CA 在可用数量较低时，它会检查这个配置条目并声明下一个范围。条目会自动更新，以便下一个 CA 获取新范围。

范围在 start *Number 和 end*Number 属性中定义，为请求和证书序列号定义单独的范围。例如：

```
dbs.beginRequestNumber=1
dbs.beginSerialNumber=1
dbs.enableSerialManagement=true
dbs.endRequestNumber=9980000
dbs.endSerialNumber=ffe0000
dbs.ldap=internaldb
dbs.newSchemaEntryAdded=true
dbs.replicaCloneTransferNumber=5
```

可以为没有克隆的 CA 启用序列号管理。 但是，除非系统被自动启用，否则将默认禁用序列号管理。

无法通过控制台手动更新序列号。序列号范围是只读字段。

- 默认签名算法.指定证书管理器用于签名证书的签名算法。如果 CA 的签名密钥类型为 RSA，则选项是 SHA256withRSA 和 SHA512withRSA。

证书配置集配置中指定的签名算法会覆盖此处设定的算法。

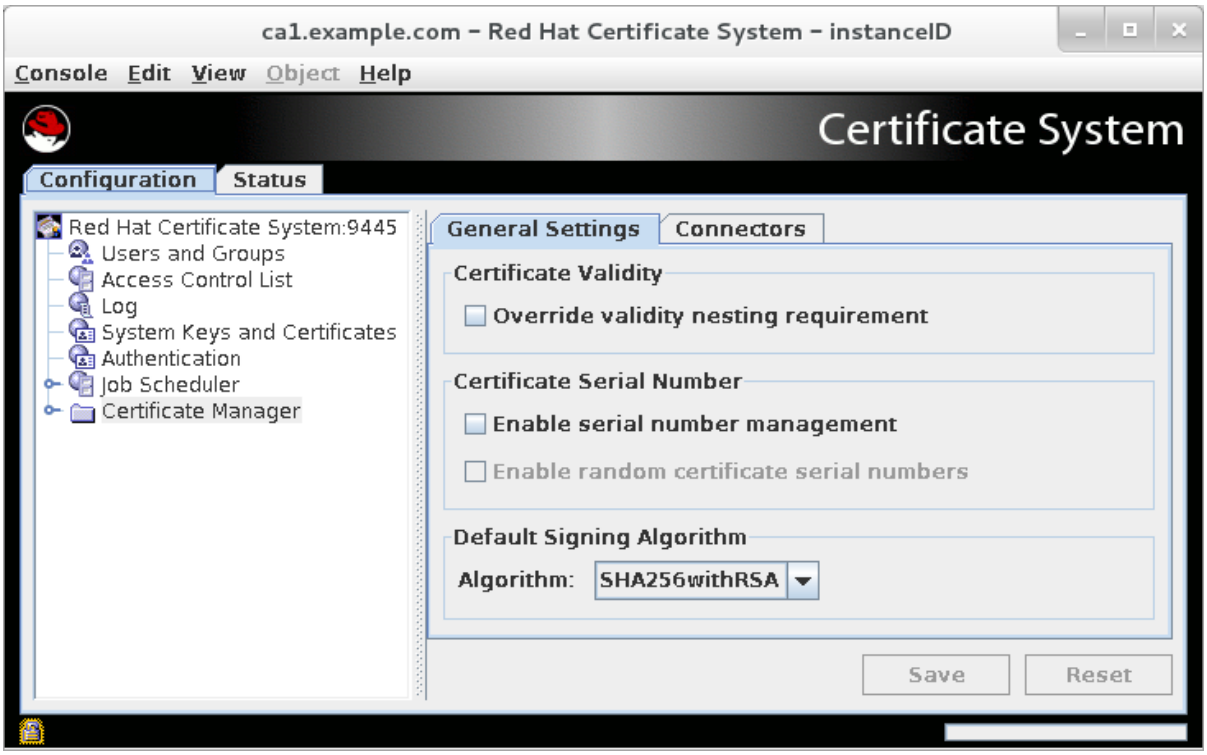
4.

默认情况下，在克隆的 CA 中，Certificate Manager 菜单项中的 General Settings 选项卡包含以下选项：

- 启用序列号管理
- 启用随机证书序列号

选中这两个复选框。

图 3.2. 默认情况下，克隆的 CA 中的 General Settings 选项卡



5.

点 Save。

3.6.3. 使用 Random 证书串行号

Red Hat Certificate System 包括一个用于请求、证书和副本 ID 的序列号范围管理。这允许在安装 Identity Management (IdM)时自动克隆。

的方法可以降低基于哈希的攻击的可能性：

- 为攻击者使证书序列号不可预测的部分
- 在身份中添加随机选择的组件
- 通过每个攻击者转发或向后兼容，使有效期日期无法预测

随机证书序列号分配 方法将随机选择的组件添加到身份。这个方法：

- 使用克隆
- 允许解决冲突
- 与当前的序列号管理方法兼容
- 与当前工作流兼容，适用于管理员、代理和最终实体
- 修复了连续序列号管理中的现有错误

备注

管理员必须启用随机证书序列号。

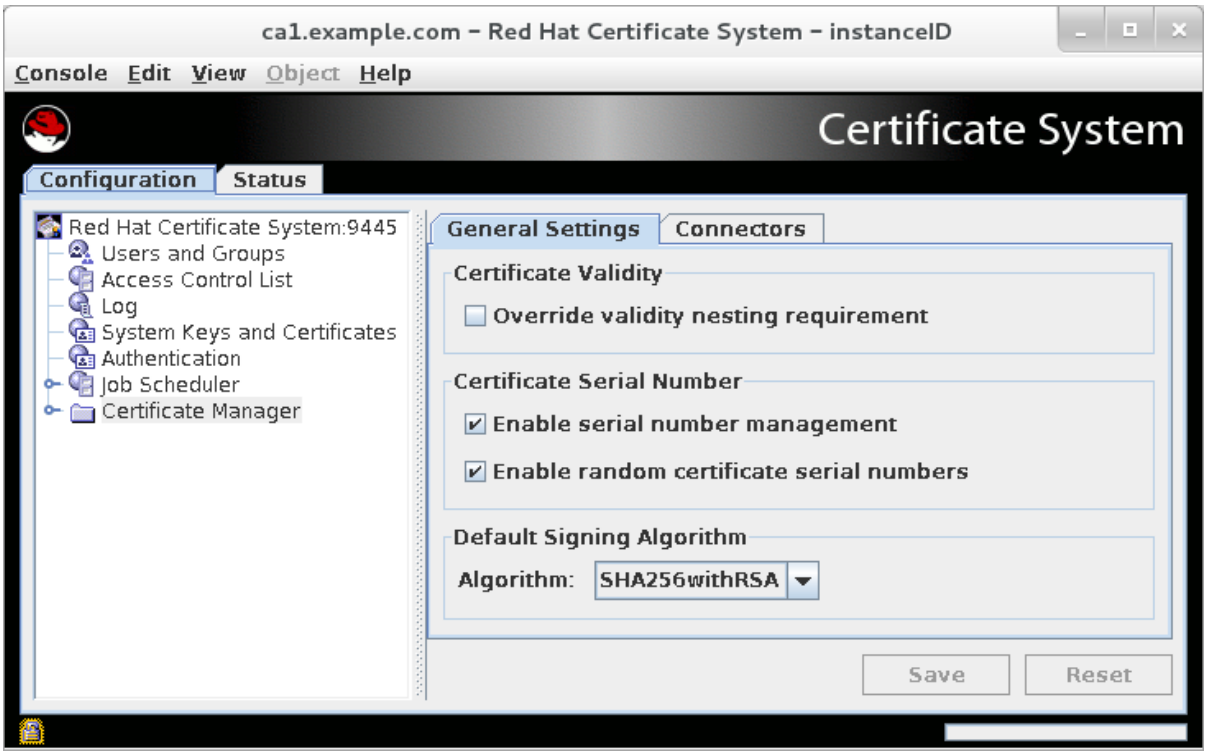
3.6.3.1. 启用 Random 证书串行数

您可以从命令行或控制台 UI 启用自动序列号管理。

从控制台 UI 启用自动序列号管理：

1. 在 General Settings 选项卡中，选择 Enable serial number 管理 选项。

图 3.3. 启用 Random Serial Number Assignment 时的 General Settings 选项卡



2. *选择 启用随机证书序列号 选项。*

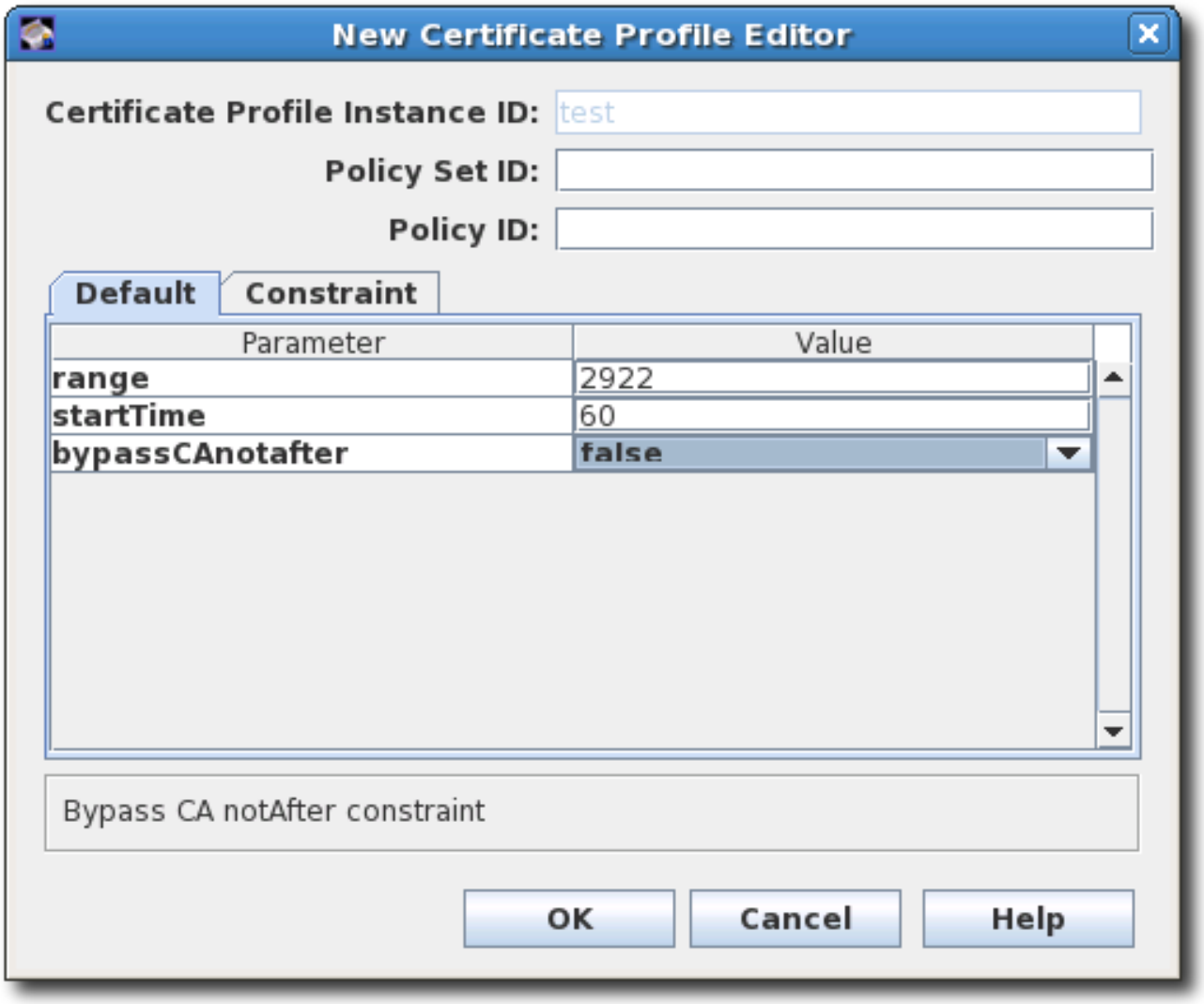
3.6.4. 允许 CA 证书续订 CA 有效期期

通常，证书无法提供在发出 CA 证书过期后终止的有效周期。如果 CA 证书具有 2015 年 12 月 31 日的过期日期，则其问题的所有证书都必须于 2015 年 12 月 31 日前过期。

这个规则适用于 CA 发布的其他 CA 签名证书，这几乎无法续订 root CA 证书。续订 CA 签名证书意味着它必须在自己的过期日期后具有有效期限。

可以使用 CA Validity Default 更改此行为。此默认设置(bypassCAnotafter)允许发布 CA 证书，具有延长发布 CA 过期时间（未经过后期）的有效周期。

图 3.4. CA 有效期默认配置



在实际部署中，这意味着 root CA 的 CA 证书可以被续订，否则可能会阻止它。

启用 CA 证书续订超过原始 CA 的有效性日期：

1. *打开 caCACert.cfg 文件。*

```
vim /var/lib/pki/instance_name/ca/conf/caCACert.cfg
```

2. *默认情况下，应存在 CA 有效期默认值。将值设为 true 以允许 CA 证书续订超过发出 CA 的有效性期。*

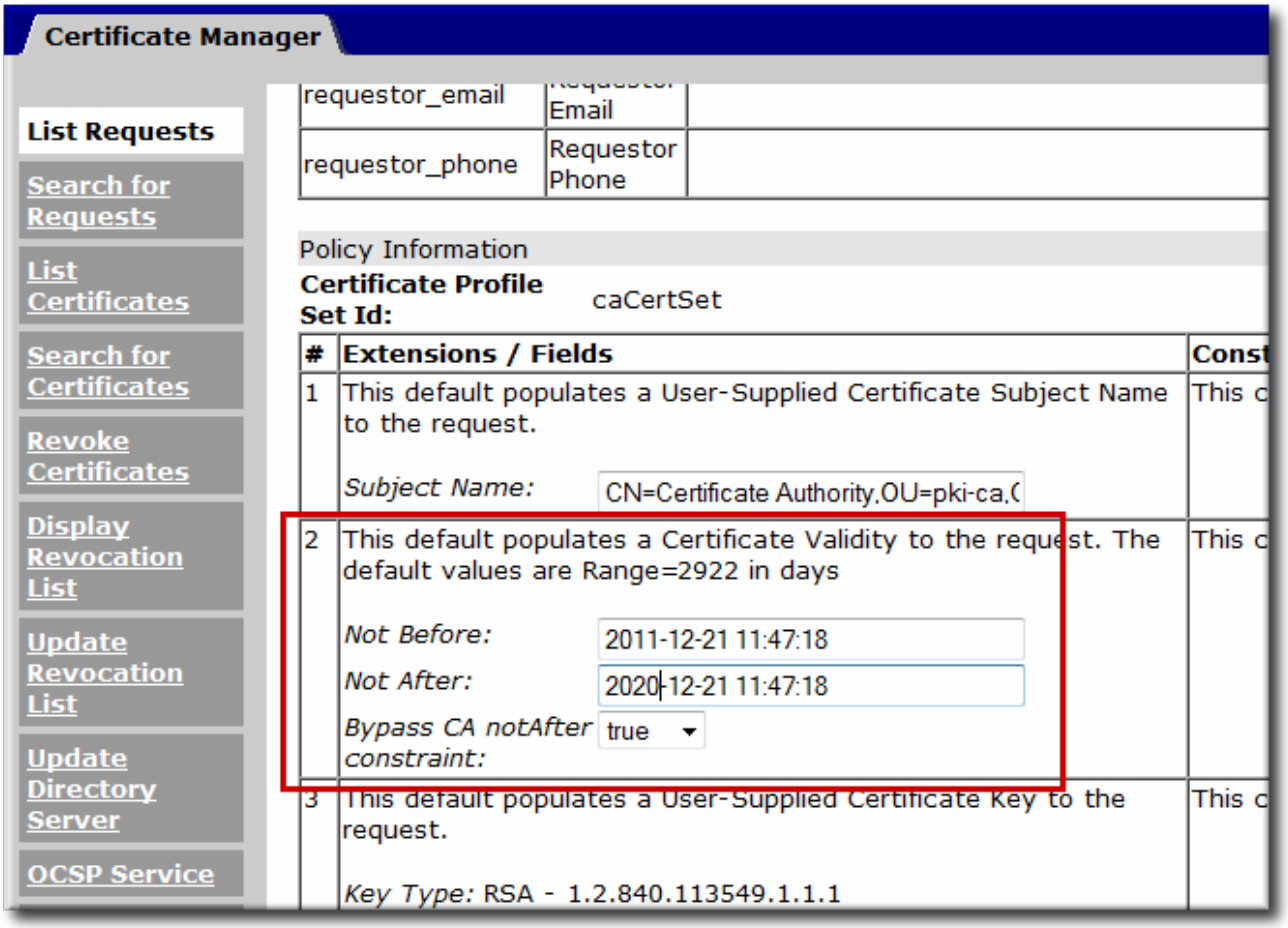
```
policyset.caCertSet.2.default.name=CA Certificate Validity Default
policyset.caCertSet.2.default.params.range=2922
policyset.caCertSet.2.default.params.startTime=0
```

```
policyset.caCertSet.2.default.params.bypassCAnotafter=true
```

3. 重启 CA 以应用更改。

当代理检查续订请求时，Extensions/Fields 区域中有一个选项，允许代理选择绕过正常的有效期约束。如果代理选择 false，则约束会被强制使用，即使配置集中设置了 bypassCAnotafter=true。如果没有启用 bypassCAnotafter 值时代理选择 true，则 CA 将拒绝续订请求。

图 3.5. 绕过代理服务页面中的 CA 限制选项



注意

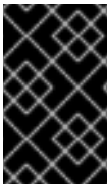
CA 有效期默认仅适用于 CA 签名证书续订。其他证书仍必须在 CA 有效期内发布和更新。

CA(ca.enablePastCATime)的独立配置设置可用于允许在 CA 有效期内续订证书。但是，这适用于该 CA 发布的 每个 证书。由于潜在的安全问题，在生产环境中不建议使用此设置。

3.7. 管理主题名称和主题备用名称

证书的 主题名称 是一个可分辨的名称(DN)，其中包含标识签发证书的实体的信息。此主题名称可从标准 LDAP 目录组件（如通用名称和组织单元）构建。这些组件在 X.500 中定义。除了 - 甚至是放置 - 主题名称，证书还可具有 主题备用名称，这是为证书设置的一种扩展集，其中包含未在 X.500 中定义的额外信息。

可以自定义主题名称和主题备用名称的命名组件。



重要

如果主题名称为空，则主题备用名称必须存在并标记为"关键"。

3.7.1. 在 Subject Name 中使用 Requester CN 或 UID

证书请求中的 cn 或 uid 值可以用来构建签发的证书的主题名称。本节演示了一个配置集，它需要在 Subject Name Constraint 中指定 naming 属性（CN 或 UID）。如果缺少 naming 属性，请求将被拒绝。

这个配置有两个部分：

- CN 或 UID 格式在 Subject Name Constraint 中的 模式 配置中设定。
- 主题 DN 的格式（包括 CN 或 UID 令牌）和证书的特定后缀在 Subject Name Default 中设定。

例如，在主题 DN 中使用 CN：

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=CN=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=CN=$request.req_subject_name.cn$,DC=example,
DC=com
```

在本例中，如果一个请求进入了 CN of cn=John Smith，则证书将使用 cn=John Smith,DC=example, DC=com 的主题 DN 颁发。如果请求来自 uid=jsmith 且无 CN，则请求将被拒绝。

相同的配置用于将请求者 UID 拉取到主题 DN 中：

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=UID=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=UID=$request.req_subject_name.uid$,DC=example,
DC=com
```

pattern 参数的格式包括在 第 B.2.11 节 “主题名称约束” 和 第 B.1.27 节 “主题名称默认” 中。

3.7.2. 在 Subject Alt Name 中插入 LDAP 目录属性值和其他信息

LDAP 目录或请求者提交的信息可通过使用 Subject Alt Name Extension Default 配置中匹配的变量插入到证书的主题备用名称中。此默认值设定信息的类型（格式），然后设置用于检索信息的匹配模式（变量）。例如：

```
policyset.userCertSet.8.default.class_id=subjectAltNameExtDefaultImpl
policyset.userCertSet.8.default.name=Subject Alt Name Constraint
policyset.userCertSet.8.default.params.subjAltNameExtCritical=false
policyset.userCertSet.8.default.params.subjAltExtType_0=RFC822Name
policyset.userCertSet.8.default.params.subjAltExtPattern_0=$request.requestor_email$
policyset.userCertSet.8.default.params.subjAltExtGNEnable_0=true
```

这会在 subject alt name 中插入请求者的电子邮件作为第一个 CN 组件。要使用额外的组件，请在数字上递增 Type_、Pattern_ 和 Enable_ 值，如 Type_1。

配置 subject alt 名称也包括在 第 B.1.23 节 “主题名称扩展默认值” 中。

将 LDAP 组件插入到证书的 subject alt 名称中：

1. 插入 LDAP 属性值需要启用用户目录身份验证插件 SharedSecret。

   a. 打开 CA 控制台。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

b. 在左侧导航树中选择 **Authentication**。

c. 在 **Authentication Instance** 选项卡中，点 **Add**，再添加 **SharedSecret** 身份验证插件的实例。

d. 输入以下信息：

```
Authentication InstanceID=SharedToken
shrTokAttr=shrTok
ldap.ldapconn.host=server.example.com
ldap.ldapconn.port=636
ldap.ldapconn.secureConn=true
ldap.ldapauth.bindDN=cn=Directory Manager
password=password
ldap.ldapauth.authtype=BasicAuth
ldap.basedn=ou=People,dc=example,dc=org
```

e. 保存新的插件实例。

有关设置 **CMC** 共享令牌的详情，请参考 第 9.4.2 节 “设置 CMC 共享 Secret”。

2. **ldapStringAttributes** 参数指示身份验证插件从用户的 LDAP 条目中读取 **mail** 属性的值，并将该值放在证书请求中。当值位于请求时，可以将证书配置集策略设置为插入扩展值的值。

dnpattern 参数的格式包括在 第 B.2.11 节 “主题名称约束” 和 第 B.1.27 节 “主题名称默认” 中。

3. 要启用 **CA** 在证书扩展中插入 **LDAP** 属性值，请编辑配置集的配置文件，并为扩展插入策略 **set** 参数。例如，要在 **caFullCMCSharedTokenCert** 配置集中的 **Subject Alternative Name** 扩展中插入 **mail** 属性值，请更改以下代码：

```
policyset.setID.8.default.params.subjAltExtPattern_0=$request.auth_token.mail[0]$
```

有关编辑配置集的详情，请参考 第 3.2.1.3 节 “使用 Raw 格式编辑证书配置文件”。

4. 重启 **CA**。

```
systemctl restart pki-tomcatd-nuxwdog@instance_name.service
```

在本例中，通过 caFullCMCSharedTokenCert 配置集注册表单提交的证书会添加 Subject Alternative Name 扩展，并带有 requester 的邮件 LDAP 属性的值。例如：

```
Identifier: Subject Alternative Name - 2.5.29.17
    Critical: no
    Value:
    RFC822Name: jsmith@example.com
```

有许多属性可自动插入到证书中，在策略集中的任何 Pattern_ 参数中设置为令牌(X)。常见令牌在表 3.1 “用于填充证书的变量” 中列出，默认配置集包含如何使用这些令牌的示例。

表 3.1. 用于填充证书的变量

策略设置令牌	描述
$request.auth_token.cn[0]$	请求证书的用户的 LDAP 通用名称(**cn**)属性。
$request.auth_token.mail[0]$	请求证书的用户的 LDAP 电子邮件（**邮件**）属性值。
$request.auth_token.tokencertsubject$	证书主题名称。
$request.auth_token.uid$	请求证书的用户的 LDAP 用户 ID(**uid**)属性。
$request.auth_token.userdn$	请求证书的用户 DN。
$request.auth_token.userid$	请求证书的用户的用户 ID 属性的值。
$request.uid$	请求证书的用户的用户 ID 属性的值。
$request.requestor_email$	提交请求的人的电子邮件地址。
$request.request_name$	提交请求的人员。
$request.upn$	Microsoft UPN。它的格式为 *(UTF8String)1.3.6.1.4.1.311.20.2.3，$request.upn$*。
$server.source$	指示服务器在主题名称中生成版本 4 个 UUID（随机号）组件。这始终具有格式 *(IA5String)1.2.3.4,$server.source$*.
$request.auth_token.user$	当请求由 TPS 提交时使用。请求证书的 TPS 子系统可信管理器。

策略设置令牌	描述
$request.subject$	当请求由 TPS 提交时使用。TPS 已解析和请求的实体的主题名称 DN。例如： **cn=John.Smith.123456789,o=TMS Org**

3.7.3. 使用 SAN 扩展中的 CN 属性

有几个客户端应用程序和库不再支持使用 Subject DN 的 Common Name(CN)属性进行域名验证，在 RFC 2818 中已被弃用。相反，这些应用程序和库在证书请求中使用 dNSName Subject Alternative Name(SAN)值。

只有在根据 RFC 1034 第 3.5 节 3.5 节 和具有多个组件匹配首选名称语法时，证书系统才会复制 CN。另外，现有 SAN 值会被保留。例如，基于 CN 的 dNSName 值被附加到现有 SAN。

要将证书系统配置为自动使用 SAN 扩展中的 CN 属性，请编辑用于发布证书的证书配置集。例如：

1. *禁用配置集：*

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-disable profile_name
```

2. *编辑配置集：*

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-edit profile_name
```

   a. *添加带有配置集唯一设置号的以下配置。例如：*

```
policyset.serverCertSet.12.constraint.class_id=noConstraintImpl
policyset.serverCertSet.12.constraint.name=No Constraint
policyset.serverCertSet.12.default.class_id=commonNameToSANDefaultImpl
policyset.serverCertSet.12.default.name=Copy Common Name to Subject
```

   前面的示例使用 12 作为集合号。

   b. *将新策略设置号附加到 policyset.userCertSet.list 参数。例如：*

```
policyset.userCertSet.list=1,10,2,3,4,5,6,7,8,9,12
```

c. 保存配置集。

3. 启用配置集：

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-enable profile_name
```

注意

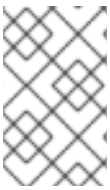
所有默认服务器配置集都包含 commonNameToSANDefaultImpl 默认。

3.7.4. 接受 CSR 的 SAN 扩展

在某些情况下，管理员希望在证书签名请求(CSR)中指定 Subject Alternative Name(SAN)扩展。

3.7.4.1. 将配置文件配置为来自 CSR 的检索 SAN

要允许从 CSR 检索 SAN，请使用用户扩展默认值。详情请查看 第 B.1.32 节 “用户补充默认设置”。



注意

SAN 扩展可以包含一个或多个 SAN。

要接受来自 CSR 的 SAN，请在配置集中添加以下默认设置和约束，如 caCMCECserverCert ：

```
prefix.constraint.class_id=noConstraintImpl
prefix.constraint.name=No Constraint

prefix.default.class_id=userExtensionDefaultImpl
prefix.default.name=User supplied extension in CSR
prefix.default.params.userExtOID=2.5.29.17
```

3.7.4.2. 使用 SAN 生成 CSR

例如，使用 certutil 实用程序通过两个 SAN 生成 CSR：

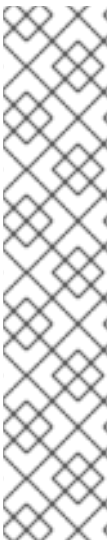
```
# certutil -R -k ec -q nistp256 -d . -s "cn=Example Multiple SANs" --extSAN
dns:www.example.com,dns:www.example.org -a -o /root/request.csr.p10
```

生成 CSR 后，请按照 第 5.6.2 节 “CMC 注册过程” 中描述的步骤完成 CMC 注册。

第 4 章 设置密钥存档和恢复

有关密钥存档和恢复的更多信息，请参阅 Red Hat 证书系统规划 、安装和部署指南中的存档激活、恢复和轮转密钥 部分。

本章论述了如何设置密钥恢复授权(KRA)，之前被称为 Data Recovery Manager(DRM)归档私钥并恢复用于恢复加密数据的归档密钥。



注意

本章仅讨论通过客户端密钥生成归档密钥。此处不讨论服务器端密钥生成和归档（无论是通过 TPS 启动还是通过 CA 的终止实体门户启动）。

有关智能卡密钥恢复的详情，请参考 第 6.11 节 “设置服务器端密钥生成”。

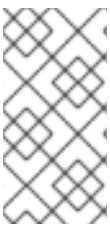
有关在 CA 的 EE 门户中提供的服务器端密钥生成的详情，请参考 第 5.2.2 节 “使用服务器侧密钥生成 CSR”。

注意

Gemalto SafeNet LunaSA 只支持 CKE - 密钥导出模型中的 PKI 私钥提取，且仅在非 FIPS 模式中支持。LunaSA Cloning 模型和 FIPS 模式中的 CKE 模型不支持 PKI 私钥提取。

安装 KRA 后，它会加入安全域，并与 CA 对。因此，它被配置为归档和恢复私钥。但是，如果 KRA 证书由外部 CA 而不是安全域中的其中一个 CA 发布，则必须手动设置密钥存档和恢复过程。

如需更多信息，请参阅 红帽认证系统规划、安装和部署指南中的 手动设置密钥存档 部分。



注意

在克隆的环境中，需要手动设置密钥归档和恢复。如需更多信息，请参阅 Red Hat 证书系统规划、安装和部署指南中的 更新 CA-KRA Connector Information 部分。

4.1. 在控制台中配置代理验证密钥恢复

注意

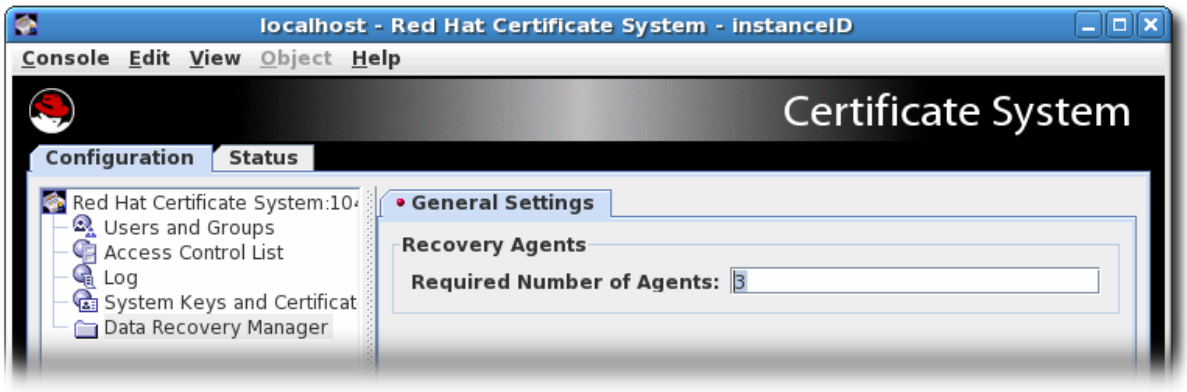
虽然可以在 控制台中配置密钥恢复代理 数量，而 要使用的组 只能直接在 CS.cfg 文件中设置。默认情况下，控制台使用密钥恢复授权代理组。

1. 打开 KRA 的控制台。例如：

```
pkiconsole https://server.example.com:8443/kra
```

2. 点左侧导航树中的 Key Recovery Authority 链接。

3. 在 所需的代理数量字段中，输入用来批准密钥恢复的代理数量。



注意

有关如何在 CS.cfg 文件中配置代理批准的密钥恢复的更多信息，请参阅 红帽证书系统规划、安装和部署指南中的 "配置代理 -Approved Key Recovery "部分。

4.2. 测试密钥存档和恢复设置

注意

较新的浏览器不支持浏览器的关键存档；对于第 1 步，应该替换这些 浏览器的 CRMF 生成客户端。

测试密钥是否可以成功归档：

1. 使用 CA 的手动用户签名和加密证书注册表单注册 2 证书。

2. 提交请求。登录代理服务页面，并批准请求。

3. 登录终端页面，并查看证书是否已颁发。在证书列表中，应该有两个带有连续序列号的新证书。

4. 将证书导入 Web 浏览器。

5. 确认密钥已被存档。在 KRA 的 agent services 页面中，选择 Show completed requests。如果密钥被成功存档，则会有有关该密钥的信息。如果没有显示密钥，请检查日志并更正问题。如果密钥已成功归档，请关闭浏览器窗口。

6. 验证 密钥。发送已签名并加密的电子邮件。收到电子邮件后，请打开它并检查消息，以查看其是否已签名并加密。消息窗口右上角应当有一个安全图标，这表示消息已签名并加密。

7. 删除证书。再次检查加密的电子邮件；邮件客户端应该无法解密邮件。

8. 测试归档的密钥是否可以成功恢复：

   a. 打开 KRA 的代理服务页面，点 Recover Keys 链接。按照密钥所有者、序列号或公钥搜索密钥。如果密钥已成功存档，则会显示密钥信息。

   b. 点 Recover。

   c. 在出现的表单中，输入与私钥对应的 base-64 编码证书来恢复；使用 CA 获取此信息。如果通过提供 base-64 编码证书搜索归档的密钥，则不得在此处提供证书。

   d. 确保已选中 Async Recovery 复选框，以便允许在恢复期间关闭浏览器会话。

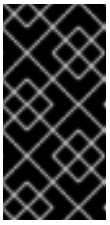
提示

async 恢复是执行密钥恢复的默认和推荐方法。如果要执行同步密钥恢复，浏览器窗口将无法关闭，且在恢复过程中无法停止 KRA。

e. *根据代理方案，指定数量的代理必须授权这个密钥恢复。让代理搜索密钥恢复，然后批准启动的恢复。*

f. *当所有代理都授权恢复后，下一屏幕将请求一个密码来加密带有证书的 PKCS #12 文件。*

g. *下一屏幕返回一个下载 PKCS #12 blob 的链接，其中包含恢复的密钥对。按照链接，将 blob 保存到文件。*



重要

在某些情况下，直接从 gcr-viewer 工具中的浏览器打开 PKCS #12 文件。要临时解决这个问题，请下载文件并在 gcr-viewer 中手动打开。

9. *将密钥恢复到浏览器的数据库。将 .p12 文件导入浏览器和邮件客户端。*

10. *打开测试电子邮件。应再次显示该消息。*

第 5 章 请求、注册和管理证书

证书由最终用户请求和使用。虽然证书注册和续订是不仅限于管理员的、了解注册和续订流程的操作，但管理员更容易管理和创建适当的证书配置文件，如 第 3.2 节 “设置证书配置集” 所述，并为每个证书类型使用适合验证方法（在 第 9 章 注册证书的身份验证中进行。

本章讨论请求、接收和更新证书以用于外部证书证书；系统。有关请求和更新证书证书nbsp;System 子系统证书的详情，请参考 第 16 章 管理子系统证书。

5.1. 关于注册和续订证书

注册 是请求和获得证书的过程。注册过程的具体方法根据证书的类型、生成其密钥对的方法以及生成和批准证书本身的方法稍有不同。无论具体方法、证书注册（在高级别）都有相同的基本步骤：

1. 生成证书请求(CSR)。
2. 证书请求提交至 CA。
3. 请求通过验证请求它的实体进行验证，确认请求是否满足用于提交它的证书配置文件规则。
4. 申请已批准。
5. 请求方检索新证书。

当证书达到其有效期期结束时，可以续订。

5.2. 创建证书签名请求

通常，以下方法用于生成证书请求(CSR)：

- 使用命令行工具生成 CSR

- 在支持的浏览器中生成 CSR
- 在应用程序内生成 CSR，如服务器的安装程序

其中一些方法支持直接提交 CSR，而有些方法则不支持。

从 RHCS 9.7 开始，支持 Server-Side 键生成来克服在较新版本的浏览器（如 Firefox v69 和 up）中消除密钥生成带来的不便性。因此，在本节中，我们将不讨论密钥生成浏览器支持。虽然没有原因认为这些浏览器的旧版本不应与旧的 RHCS 文档中所述继续工作。

从应用程序生成的 CSR 通常采用 PKCS#10 的形式。如果它们正确生成，则 RHCS 应该支持它们。

在以下小节中，我们将深入介绍 RHCS 支持的以下方法：

- 命令行工具
- 服务器侧密钥生成

5.2.1. 使用命令行工具生成 CSR

Red Hat Certificate System 支持使用以下工具创建 CSR：

- certutil ：支持创建 PKCS #10 请求。
- PKCS10Client ：支持创建 PKCS #10 请求。
- CRMFPopClient ：支持创建 CRMF 请求。
- pki client-cert-request: 支持 PKCS#10 和 CRMF 请求。

以下小节介绍了如何将这些实用程序与功能丰富的注册配置集框架搭配使用的一些示例。

5.2.1.1. 使用 certutil创建 CSR

这部分论述了如何使用 certutil 工具创建 CSR 的示例。

有关使用 certutil 的详情，请参考：

- certutil(1) man page
- certutil --help 命令的输出

5.2.1.1.1. 使用 certutil 创建带有 EC 密钥的 CSR

以下流程演示了如何使用 certutil 实用程序创建 Elliptic Curve(EC)密钥对和 CSR：

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 创建二进制 CSR，并将其存储在 /user_or_entity_database_directory/request.csr 文件中：

```
$ certutil -d . -R -k ec -q nistp256 -s "CN=subject_name" -o
/user_or_entity_database_directory/request-bin.csr
```

提示时输入所需的 NSS 数据库密码。

有关参数的详情，请查看 certutil(1) man page。

3. 将创建的二进制格式 CSR 转换为 PEM 格式：

```
$ BtoA /user_or_entity_database_directory/request-bin.csr
/user_or_entity_database_directory/request.csr
```

4. （可选）验证 CSR 文件是否正确：

```
$ cat /user_or_entity_database_directory/request.csr

MIICbTCCAVUCAQAwKDEQMA4GA1UEChMHRXhhbXBsZTEUMBIGA1UEAxMLZXhhbXBs
...
```

这是一个 PKCS#10 PEM 证书请求。

5.2.1.1.2. 使用 certutil 使用用户定义的扩展来创建 CSR

以下流程演示了如何使用 certutil 实用程序使用用户定义的扩展创建 CSR。

请注意，注册请求受 CA 定义的注册配置集的限制。请参阅 例 B.3 “CSR 中的多个用户提供的扩展”。

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 使用用户定义的 Key Usage 扩展创建 CSR，以及用户定义的扩展密钥使用扩展，并将其存储在 /user_or_entity_database_directory/request.csr 文件中：

```
$ certutil -d . -R -k rsa -g 1024 -s "CN=subject_name" --keyUsage
keyEncipherment,dataEncipherment,critical --extKeyUsage
timeStamp,msTrustListSign,critical -a -o /user_or_entity_database_directory/request.csr
```

提示时输入所需的 NSS 数据库密码。

有关参数的详情，请查看 certutil(1) man page。

3. （可选）验证 CSR 文件是否正确：

```
$ cat /user_or_entity_database_directory/request.csr
Certificate request generated by NSS certutil
Phone: (not specified)

Common Name: user 4-2-1-2
Email: (not specified)
Organization: (not specified)
State: (not specified)
Country: (not specified)
```

这是一个 PKCS#10 PEM 证书请求。

5.2.1.2. 使用 PKCS10Client创建 CSR

这部分论述了如何使用 PKCS10Client 实用程序创建 CSR 的示例。

有关使用 PKCS10Client 的详情，请参考：

- PKCS10Client(1) man page
- PKCS10Client --help 命令的输出

5.2.1.2.1. 使用 PKCS10Client 创建 CSR

以下流程解释了如何使用 PKCS10Client 实用程序创建 Elliptic Curve(EC)密钥对和 CSR：

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 创建 CSR，并将其存储在 /user_or_entity_database_directory/example.csr 文件中：

```
$ PKCS10Client -d . -p NSS_password -a ec -c nistp256 -o
/user_or_entity_database_directory/example.csr -n "CN=subject_name"
```

有关参数的详情，请查看 PKCS10Client(1) man page。

3. （可选）验证 CSR 是否正确：

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.2.2. 使用 PKCS10Client 为基于 SharedSecret 的 CMC 创建 CSR

以下流程解释了如何使用 PKCS10Client 程序为基于 SharedSecret 的 CMC 创建 RSA 密钥对和 CSR。它仅与 CMC Shared Secret 身份验证方法一起使用，默认由 caFullCMCSharedTokenCert 和 caECFullCMCSharedTokenCert 配置集处理。

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 创建 CSR，并将其存储在 /user_or_entity_database_directory/example.csr 文件中：

```
$ PKCS10Client -d . -p NSS_password -o /user_or_entity_database_directory/example.csr -y true -n "CN=subject_name"
```

有关参数的详情，请查看 PKCS10Client(1) man page。

3. （可选）验证 CSR 是否正确：

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.3. 使用 CRMFPopClient创建 CSR

证书请求消息格式(CRMF)是 CMC 中接受的 CSR 格式，允许密钥存档信息安全地嵌入到请求中。

本节介绍如何使用 CRMFPopClient 实用程序创建 CSR 的示例。

有关使用 CRMFPopClient 的详情，请查看 CRMFPopClient(1) man page。

5.2.1.3.1. 使用 CRMFPopClient 创建带有密钥 Archival 的 CSR

以下流程解释了如何使用 CRMFPopClient 实用程序创建 RSA 密钥对，并使用密钥归档选项创建 CSR：

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 检索 KRA 传输证书：

```
$ pki ca-cert-find --name "DRM Transport Certificate"
 ---------------
 1 entries found
 ---------------
  Serial Number: 0x7
  Subject DN: CN=DRM Transport Certificate,O=EXAMPLE
  Status: VALID
  Type: X.509 version 3
  Key A    lgorithm: PKCS #1 RSA with 2048-bit key
  Not Valid Before: Thu Oct 22 18:26:11 CEST 2015
  Not Valid After: Wed Oct 11 18:26:11 CEST 2017
  Issued On: Thu Oct 22 18:26:11 CEST 2015
  Issued By: caadmin
 ----------------------------
 Number of entries returned 1
```

3. 导出 KRA 传输证书：

```
$ pki ca-cert-show 0x7 --output kra.transport
```

4. 创建 CSR，并将其存储在 /user_or_entity_database_directory/example.csr 文件中：

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b
kra.transport -w "AES/CBC/PKCS5Padding" -v -o
/user_or_entity_database_directory/example.csr
```

要创建 Elliptic Curve(EC)密钥对和 CSR，将 -a ec -t false 选项传递给该命令。

有关参数的详情，请查看 CRMFPopClient(1) man page。

5. （可选）验证 CSR 是否正确：

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.3.2. 使用 CRMFPopClient 为基于 SharedSecret 的 CMC 创建 CSR

以下流程解释了如何使用 CRMFPopClient 程序为基于 SharedSecret 的 CMC 创建 RSA 密钥对和 CSR。它仅与 CMC Shared Secret 身份验证方法一起使用，默认由 caFullCMCSharedTokenCert 和 caECFullCMCSharedTokenCert 配置集处理。

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /user_or_entity_database_directory/
```

2. 检索 KRA 传输证书：

```
$ pki ca-cert-find --name "DRM Transport Certificate"
 ---------------
 1 entries found
 ---------------
  Serial Number: 0x7
  Subject DN: CN=DRM Transport Certificate,O=EXAMPLE
  Status: VALID
  Type: X.509 version 3
  Key A    lgorithm: PKCS #1 RSA with 2048-bit key
  Not Valid Before: Thu Oct 22 18:26:11 CEST 2015
  Not Valid After: Wed Oct 11 18:26:11 CEST 2017
  Issued On: Thu Oct 22 18:26:11 CEST 2015
  Issued By: caadmin
 ----------------------------
 Number of entries returned 1
```

3. 导出 KRA 传输证书：

```
$ pki ca-cert-show 0x7 --output kra.transport
```

4. 创建 CSR，并将其存储在 /user_or_entity_database_directory/example.csr 文件中：

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b
kra.transport -w "AES/CBC/PKCS5Padding" -y -v -o
/user_or_entity_database_directory/example.csr
```

要创建 EC 密钥对和 CSR，将 -a ec -t false 选项传递给该命令。

有关参数的详情，请查看 CRMFPopClient --help 命令的输出。

5. （可选）验证 CSR 是否正确：

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.4. 在 PKI CLI 中使用 client-cert-request 创建 CSR

pki命令行工具也可以与 client-cert-request 命令一起使用，以生成 CSR。但是，与之前讨论的工具不同，使用 pki 生成的 CSR 将直接提交到 CA。可以生成 PKCS#10 或 CRMF 请求。

生成 PKCS#10 请求的示例：

```
pki -d user token db directory -P https -p 8443 -h host.test.com -c user token db passwd client-
cert-request "uid=test2" --length 4096 --type pkcs10
```

生成 CRMF 请求的示例：

```
pki -d user token db directory -P https -p 8443 -h host.test.com -c user token db passwd client-
cert-request "uid=test2" --length 4096 --type crmf
```

成功时将返回请求 id。

提交请求后，代理可以通过使用 pki ca-cert-request-approve 命令批准它。

例如：

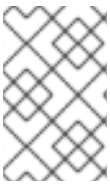
```
pki -d agent token db directory -P https -p 8443 -h host.test.com -c agent token db passwd -n <CA agent cert nickname> ca-cert-request-approve request id
```

如需更多信息，请参阅 pki client-cert-request --help 命令的 man page。

5.2.2. 使用服务器侧密钥生成 CSR

许多新版本的浏览器（包括 Firefox v69 和 up 以及 Chrome）都已删除了生成 PKI 密钥的功能，以及对 CRMF 进行密钥归档的支持。在 RHEL 上，可以使用 CLI（如 CRMFPopClient --help）或 pki（请参阅 pki client-cert-request --help）作为临时解决方案。

自引入令牌密钥管理系统(TMS)起，服务器端密钥注册时间已足够长，可在 KRA 中生成密钥，而不是在智能卡上本地生成密钥。Red Hat Certificate System 9 现在采用类似的机制来解决浏览器 keygen deficiy 问题。密钥在服务器上生成（特别是在 KRA 上），然后安全地传送回 PKCS#12 中的客户端。



注意

强烈建议您只对加密证书使用 Server-Side Keygen 机制。

5.2.2.1. 功能亮点

- 证书请求密钥在 KRA 上生成（注意：必须安装 KRA 才能使用 CA）
- 配置集默认插件 serverKeygenUserKeyDefaultImpl 提供选择来启用或禁用密钥存档（例如，enableArchival 参数）

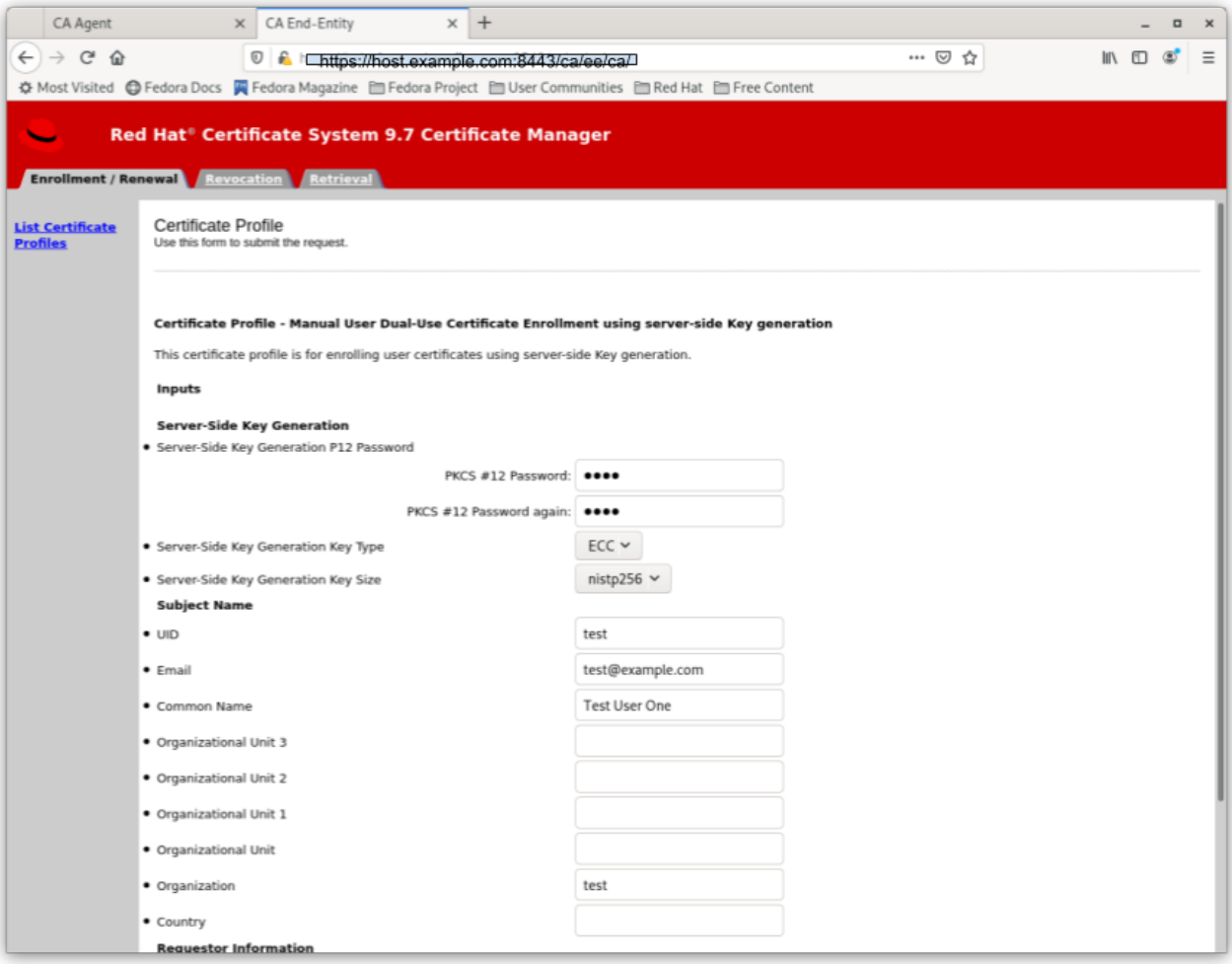
- *支持 RSA 和 EC 密钥*
- *支持手动（代理）批准和自动批准（例如，基于目录密码）*

5.2.2.2. 使用服务器侧密钥注册证书

默认的 Sever-Side Keygen 注册配置集可在 EE 页面的 List Certificate Profiles 选项卡找到：

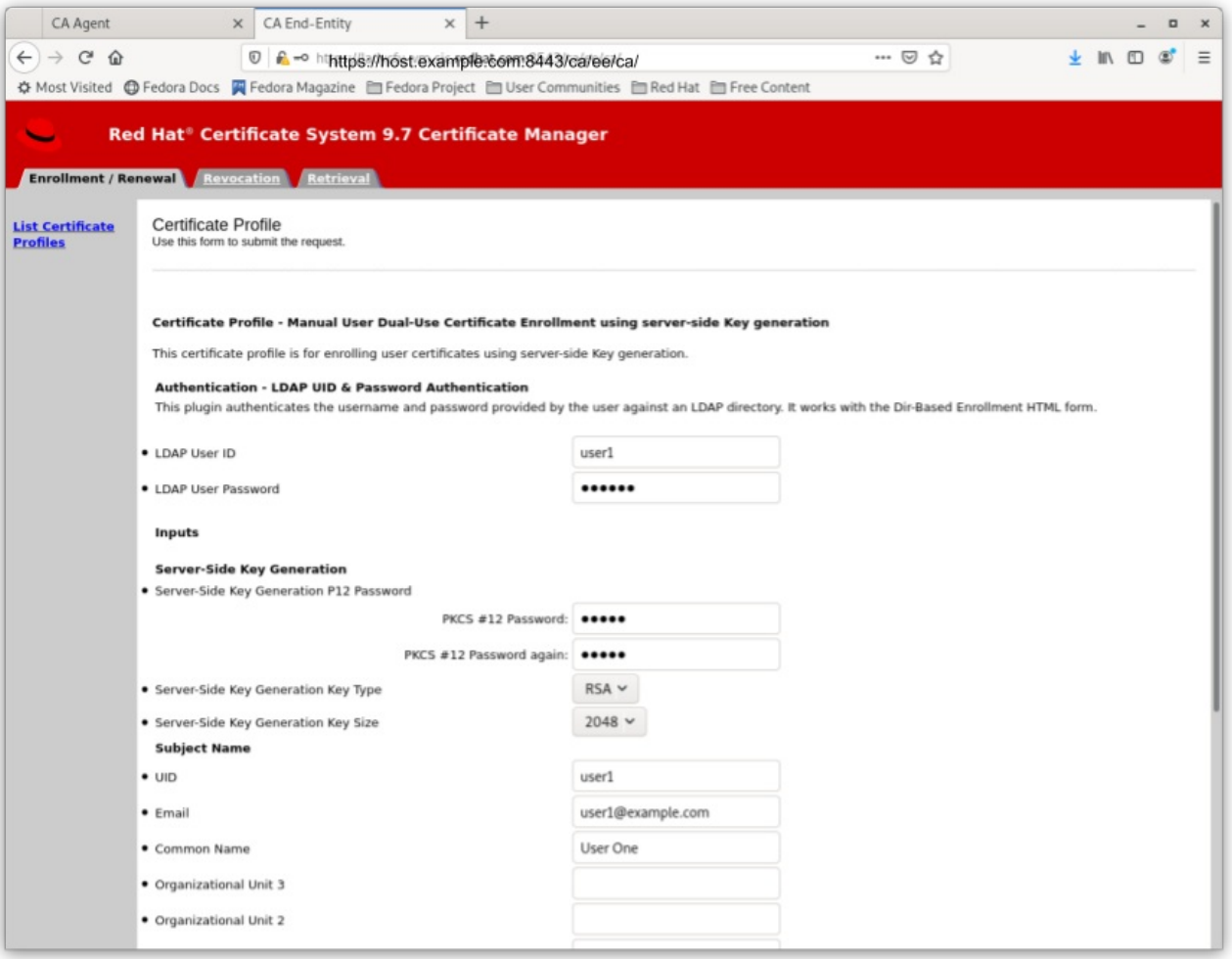
手动用户使用服务器端密钥注册注册

图 5.1. 需要代理手动批准的服务器端密钥生成



使用服务器端密钥生成目录验证的用户注册

图 5.2. server-Side Keygen Enrollment，在成功 LDAP uid/pwd 身份验证后将自动批准



无论如何批准请求，服务器侧密钥注册机制都需要最终用户输入 PKCS#12 软件包的密码，其中包含签发的证书以及服务器发布后生成的加密私钥。

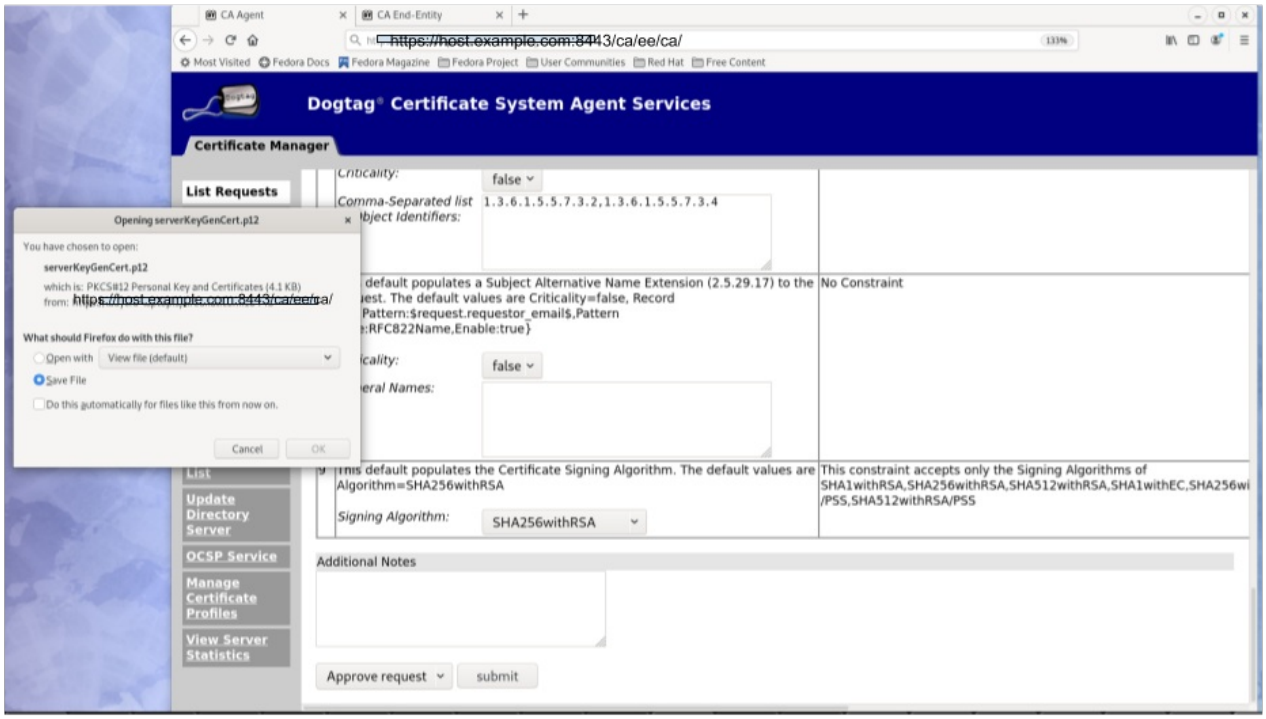
重要

用户不应与任何人共享其密码。甚至是 CA 或 KRA 代理。

当注册请求被批准后，将生成一个 PKCS#12 软件包，并生成、

- *如果是手动批准，PS PKCS#12 文件将返回到批准请求的 CA 代理；然后代理应该将 PKCS#12 文件转发给用户。*
- *如果是自动批准，PMPS PKCS#12 文件将返回到提交该请求的用户*

图 5.3. 代理手动批准注册



收到 PKCS#12 文件后，用户可以使用 pkcs12util（如 pkcs12util ）将此文件导入到每个应用程序自己的用户内部证书/密钥数据库中。例如，如 Firefox nss 数据库。

5.2.2.3. 密钥恢复

如果在证书注册配置集中将 enableArchival 参数设置为 true，则在 Server-Side Keygen 注册时将存档私钥。然后，授权 KRA 代理可以恢复归档的私钥。

5.2.2.4. 其它信息

5.2.2.4.1. KRA Request Records

注意

由于这个机制的性质，在配置集中将 enableArchival 参数设为 true 时，每个 Server-Side keygen 请求有两个 KRA 请求：

- *一个用于请求类型 asymkeyGenRequest*

不能使用 KRA 代理页面上的 List Requests 过滤此请求类型；您可以选择 Show All Requests 以查看列出的请求。

- 一个用于请求类型 恢复

5.2.2.4.2. 审计记录

如果启用，可以观察一些审计记录：

CA

- SERVER_SIDE_KEYGEN_ENROLL_KEYGEN_REQUEST
- SERVER_SIDE_KEYGEN_ENROLL_KEY_RETRIEVAL_REQUEST

KRA

- SERVER_SIDE_KEYGEN_ENROLL_KEYGEN_REQUEST_PROCESSED
- SERVER_SIDE_KEYGEN_ENROLL_KEY_RETRIEVAL_REQUEST_PROCESSED（尚未实施）

5.3. 配置 INTERNET EXPLORER 以注册证书

警告

以下流程不再被支持，仅保留供参考。这个功能已从 Internet Explorer 11 中弃用，Microsoft 已结束对 IE 10 的支持。

由于 Microsoft Windows 中的安全设置，因此请求并通过使用 Internet Explorer 的最终实体页面来请求和注册证书，需要额外的浏览器配置。必须将浏览器配置为信任 CA，然后才能访问 CA 的最终用户页面。

5.3.1. 关于关键限制和 Internet Explorer

Microsoft 使用某些加密供应商，它只支持 RSA 和 ECC 密钥的潜在密钥子集。这些内容列在 表 5.1 “供应商和密钥大小” 中。

密钥大小支持可能会影响与 Internet Explorer 搭配使用的配置集配置。配置配置集包括在 第 3 章 制作发行证书规则（证书配置文件） 中。

表 5.1. 供应商和密钥大小

算法	供应商	支持的密钥大小
ECC	Microsoft Software Key Storage Provider	• nistp256 • nistp384 • nistp521
ECC	Microsoft 智能卡存储供应商	• nistp256 • nistp384 • nistp521
RSA	Microsoft Base Cryptographic Provider	• 1024
RSA	Microsoft Strong Cryptographic Provider	• 1024 • 2048 • 3072 • 4096 • 8192
RSA	增强的 Cryptographic Provider	• 1024 • 2048 • 3072 • 4096 • 8192

算法	供应商	支持的密钥大小
RSA	Microsoft Software Key Storage Provider	• 1024 • 2048 • 3072 • 4096 • 8192

5.3.2. 配置 Internet Explorer

1. *打开 Internet Explorer。*

2. *Open Tools → Internet Options → Advanced → Security，取消选择 TLS 1.2。*

3. *导入 CA 证书链。*

   a. *为 CA 打开 unsecured end 服务页面，例如：*

   ```
   http://server.example.com:8080/ca/ee/ca
   ```

   b. *点 Retrieval 选项卡。*

   c. *点左侧菜单中的 Import CA Certificate Chain，然后选择 Download the CA certificate chain in binary form。*

   d. *提示时，保存 CA 证书链文件。*

   e. *在 Internet Explorer 菜单中，点 Tools，然后选择 Internet Options。*

   f. *打开 内容选项卡，然后单击 证书按钮。*

g. 单击 导入 按钮。在导入窗口中，浏览并选择导入的证书链。

导入过程会提示将哪些证书存储用于 CA 证书链。根据证书类型选择 Automatically，选择证书存储。

h. 导入证书链后，打开 受信任的根证书颁发机构 选项卡，以验证证书链是否已成功导入。

4. 将 Internet Explorer 配置为提示允许使用不安全的 ActiveX 控制进行脚本编写。如果不允许这样做，且最终用户会尝试在标准（非 SSL）最终用户中注册证书，Internet Explorer 将阻断这些页面。

a. 在 Internet Explorer 菜单中，点 Tools 并选择 Internet Options。

b. 打开" 安全 "选项卡，然后单击" 自定义级别 "。

c. 在 ActiveX Controls 和 Plugins 区域中，将 Initialize 和 script ActiveX 控制 的值更改为 Prompt。

5. 导入证书链后，Internet Explorer 可以访问安全的最终用户页面。打开安全网站，例如：

```
https://server.example.com:8443/ca/ee/ca
```

6. 打开结束服务页面时，可能有一个安全例外。将 CA 服务网站添加到 Internet Explorer 的受信任的站点 列表。

a. 在 Internet Explorer 菜单中，点 Tools，然后选择 Internet Options。

b. 打开 Security 选项卡，再单击 Sites，将 CA 站点添加到可信列表中。

c. 将 CA 服务页面 的安全级别 设置为 Medium-High；如果以后此安全设置的限制太强，则尝试将其重置为 Medium。

7. 打开 工具 → 兼容性视图和兼容性视图设置，然后通过添加特定站点到列表来启用 兼容性 视图设置。

8. 关闭浏览器。

要验证 Internet Explorer 是否可以用来注册，请尝试注册用户证书，如 第 5.4.1 节 “通过"最终用户"页面请求并接收证书” 所述。

5.4. 请求和接收证书

如 第 5.1 节 “关于注册和续订证书” 中所述，生成 CSR 后，需要将其提交到 CA 以提高状态。第 5.2 节 “创建证书签名请求” 中讨论的一些方法直接向 CA 提交 CSR，而有些方法需要在单独的步骤中提交 CSR，这可能由用户执行，或者由代理预签名。

在本节中，我们将讨论 RHCS CA 支持的单独提交步骤。

- 第 5.4.1 节 “通过"最终用户"页面请求并接收证书”
- 第 5.6 节 “使用 CMC 提交证书请求”

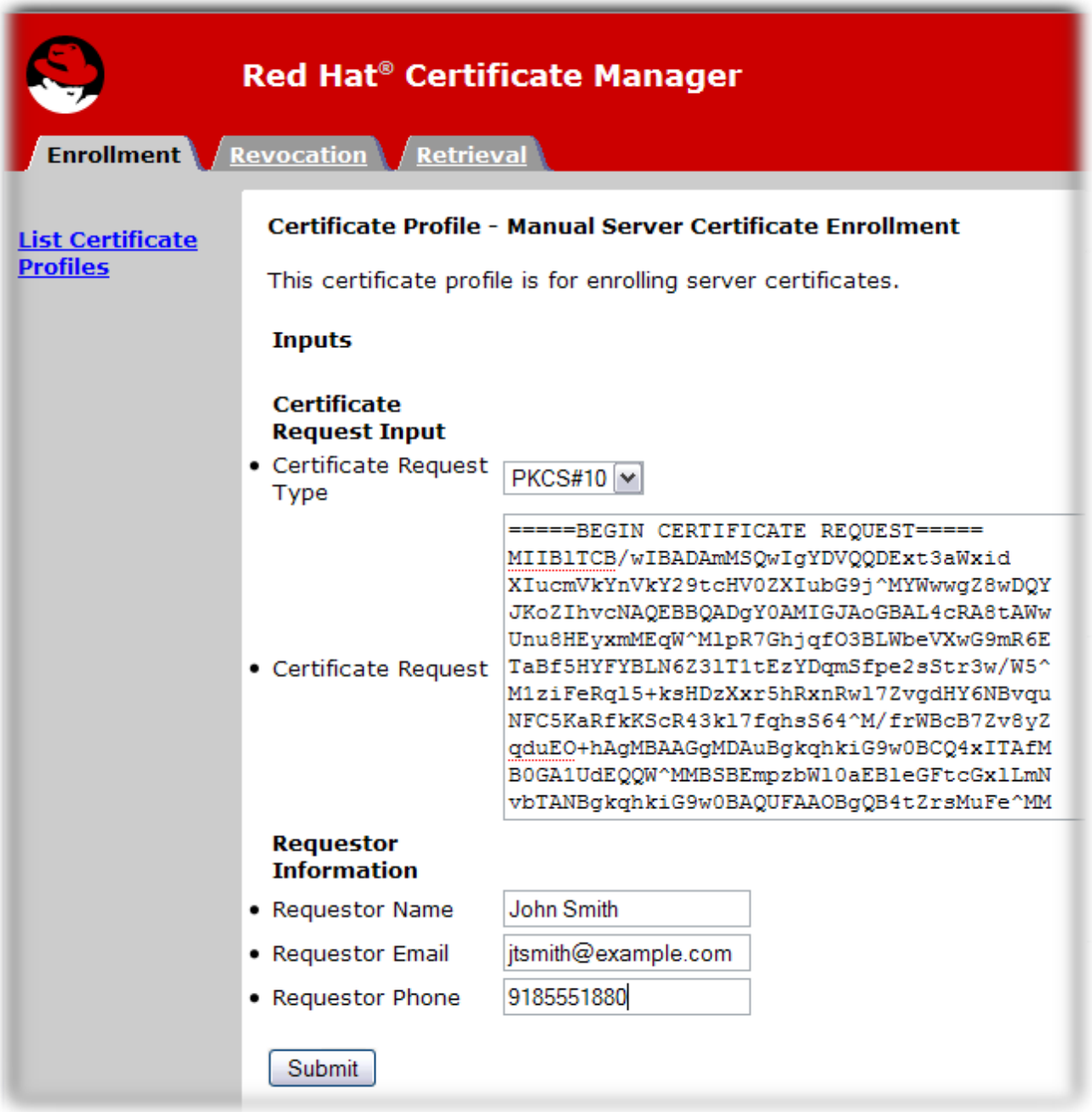
5.4.1. 通过"最终用户"页面请求并接收证书

在 CA End Entity 门户中（例如 https://host.domain:port#/ca/ee/ca），最终用户可以使用在 Enrollment/Renewal 选项卡下的 Enrollment/Renewal 选项卡下提交其证书请求（CSR，请参阅 第 5.2 节 “创建证书签名请求” 来生成 CSR）。

本节假设您具有 Base64 编码格式的 CSR，包括标记行 -----BEGIN NEW CERTIFICATE REQUEST----- 和 -----END NEW CERTIFICATE REQUEST-----。

许多默认注册配置文件都提供一个 证书请求 文本框，该框可以在 Base64 编码 CSR 中粘贴，以及证书请求类型选择下拉列表。

在证书注册表单中，输入所需信息。



标准要求如下：

- *证书请求类型.这是 PKCS#10 或 CRMF。通过子系统管理控制台创建的证书请求是 PKCS #10；通过 certutil 工具和其他实用程序创建的证书请求通常是 PKCS #10。*

- *证书请求.粘贴 base-64 编码的 blob，包括 -----BEGIN NEW CERTIFICATE REQUEST----- 和 -----END NEW CERTIFICATE REQUEST----- 标记行。*

- *申请人名.这是请求证书的人的通用名称。*

- 申请人电子邮件.这是请求者的电子邮件地址。在签发证书时，代理或 CA 系统将使用此地址联系请求者。例如 ：jdoe@someCompany.com

- 请求者电话.这是请求者的联系电话号码。

提交的请求排队代理批准。代理需要处理并批准证书请求。

注意

某些注册配置文件可能允许使用由 Red Hat Certificate Systemnbsp 提供的 LDAP uid/pwd 验证方法自动执行（例如，Red Hat Certificate Red Hat Certificate Systemnbsp;System.在下一节中，通过这些配置集进行注册不需要手动批准。有关支持的批准方法，请参阅 第 9 章 注册证书的身份验证。

如果是手动批准，一旦证书被批准并生成，您可以检索该证书。

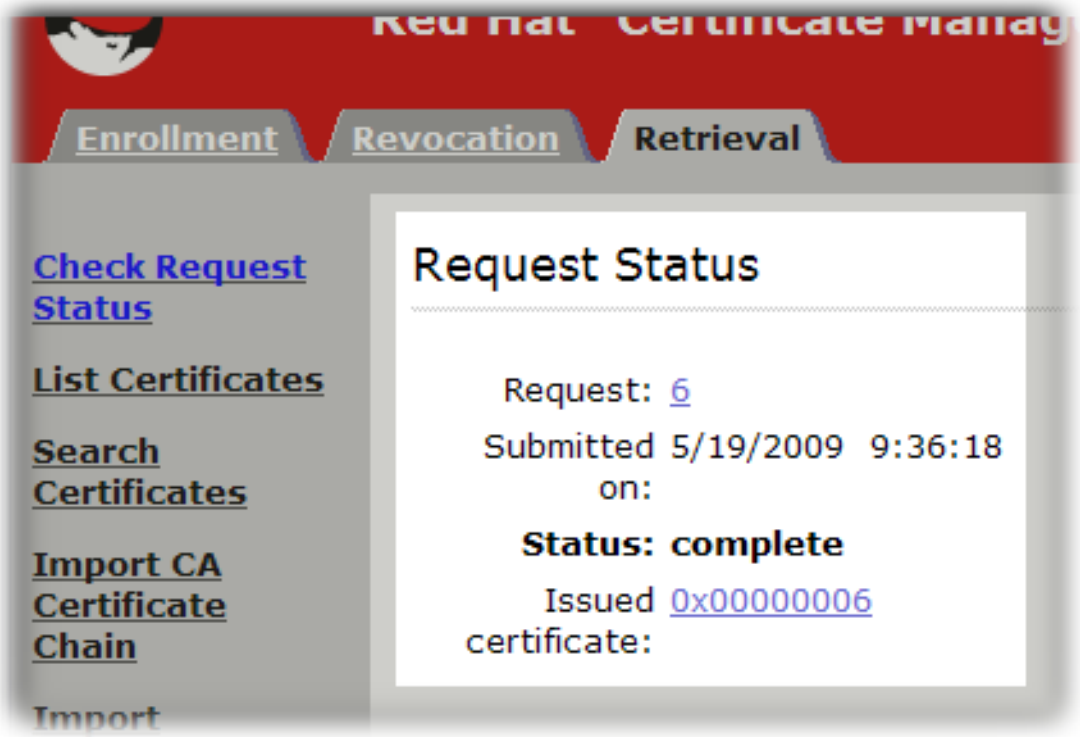
1. 打开证书管理器端点页面，例如：

```
https://server.example.com:8443/ca/ee/ca
```

2. 点 Retrieval 选项卡。

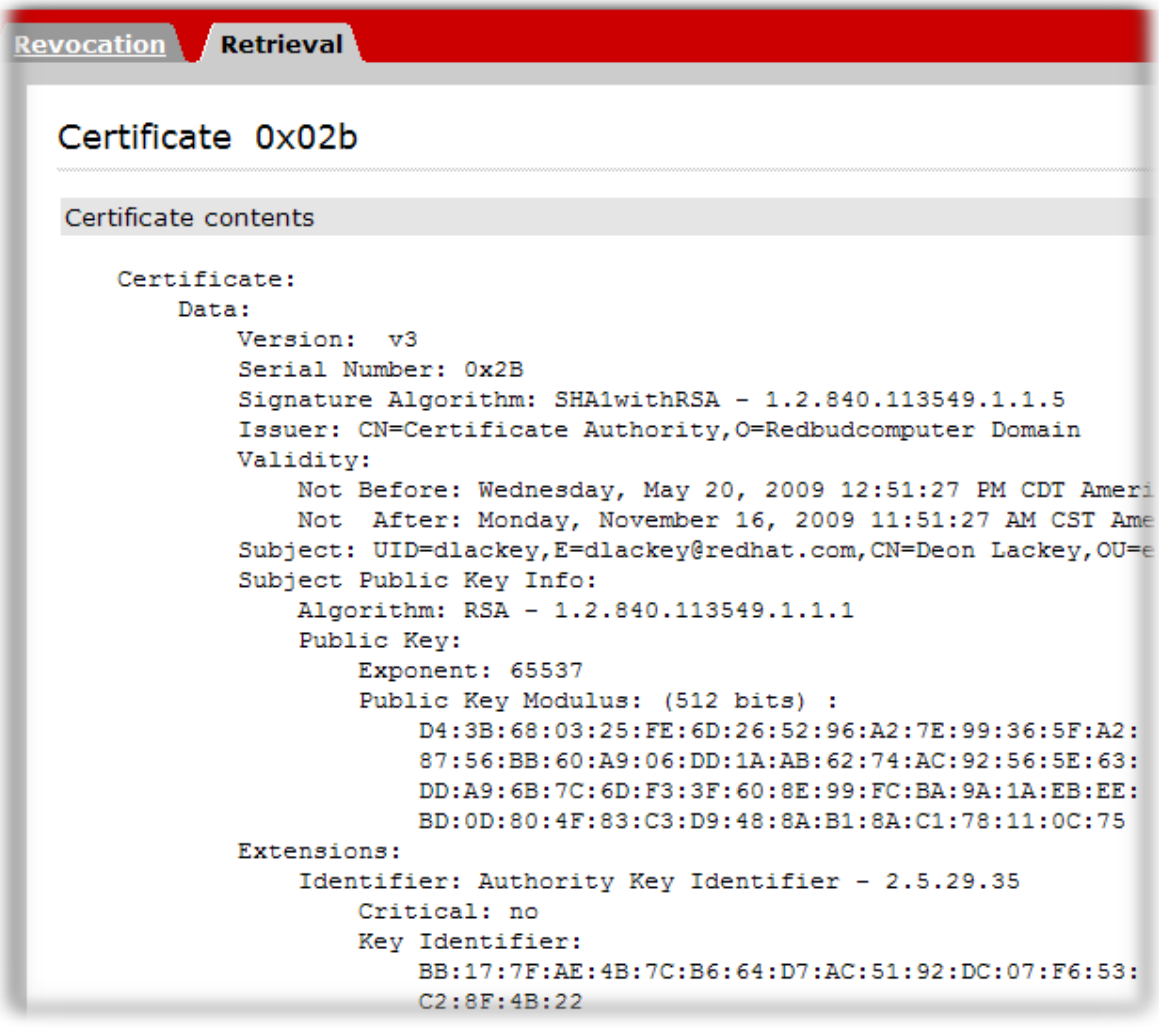
3. 填写提交证书请求时创建的请求 ID 号，然后单击 Submit。

4. 下一页显示证书请求的状态。如果状态 已经完成，则证书有一个链接。点 签发的证书 链接。



5.

新证书信息以用户用户打印格式显示，采用 base-64 编码格式，采用 PKCS #7 格式。



可以通过此页面执行以下操作：

- 要在服务器或其他应用程序上安装此证书，请滚动到"服务器"部分中安装此证书，其中包含 base-64 编码证书。

6. 将 base-64 编码的证书（包括 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE----- 标记行）复制到文本文件。保存文本文件，并使用它来将证书的副本存储在私钥所驻留的实体的安全模块中。请参阅 第 14.3.2.1 节 “创建用户”。

5.5. 续订证书

本节讨论如何更新证书。有关如何设置证书续订的详情请参考 第 3.4 节 “配置配置集以启用续订”。

续订证书包括重新生成证书，其属性与原始证书相同。通常，有两个类型的续订：

- 相同的密钥续订 是原始密钥、配置文件和请求证书，并使用相同密钥重新创建具有新有效期期限和到期日期的新证书。这可以通过以下任一方法完成：
  - 通过原始配置集(CSR)重新提交原始证书请求(CSR)，或者
  - 使用支持工具（如 certutil）使用原始密钥重新生成 CSR
- 重新标记证书需要使用相同的信息重新生成证书请求，以便生成新的密钥对。然后，CSR 通过原始配置集提交。

5.5.1. 相同的密钥续订

5.5.1.1. 重新使用 CSR

在终端实体门户上，有三种批准方法可以进行相同的密钥续订。

- agent-approved 方法需要提交要续订的证书的序列号；此方法需要 CA 代理的批准。

- 基于目录的续订需要提交要续订的证书的序列号，并且 CA 从其当前证书目录条目中提取信息。如果 ldap uid/pwd 已被成功进行身份验证，则证书会被自动批准。

- 基于证书的续订使用浏览器数据库中的证书进行身份验证，并具有重新发布相同的证书。

5.5.1.1.1. 代理(Approved)或基于目录的续订

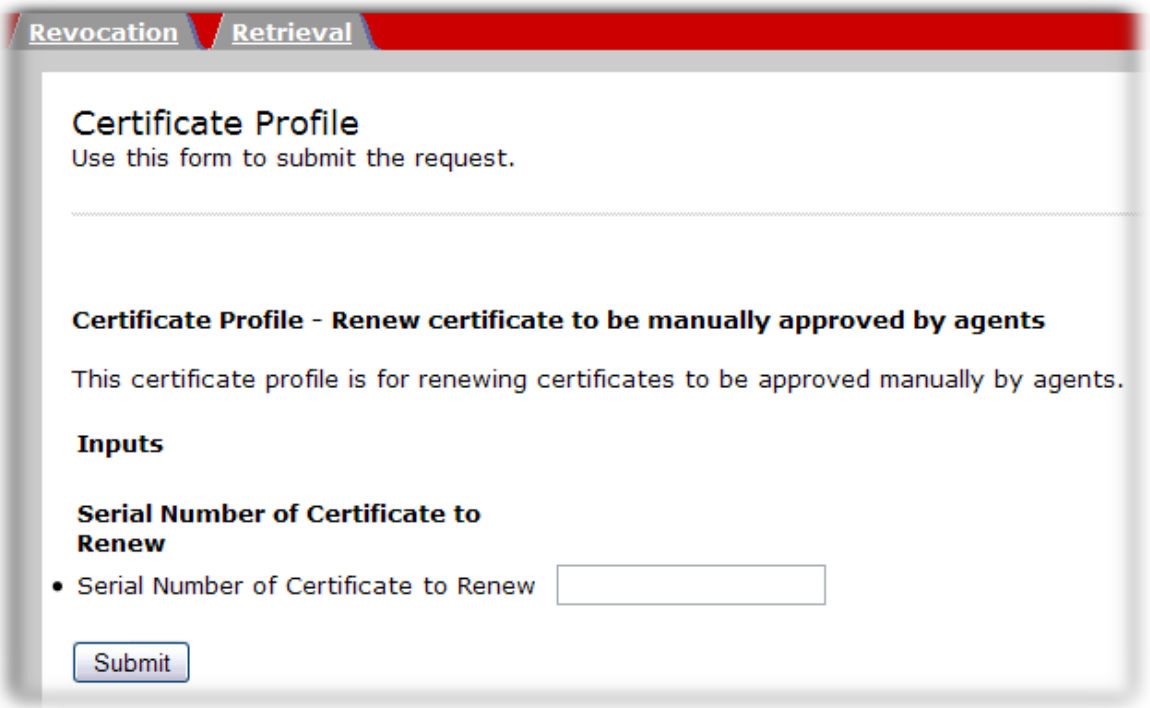
有时，必须手动批准证书续订请求，可以是 CA 代理，或者提供用户目录的登录信息。

1. 打开发布证书的 CA 的端点服务页面（或其克隆）。

   https://server.example.com:8443/ca/ee/ca

2. 单击要使用的续订表单的名称。

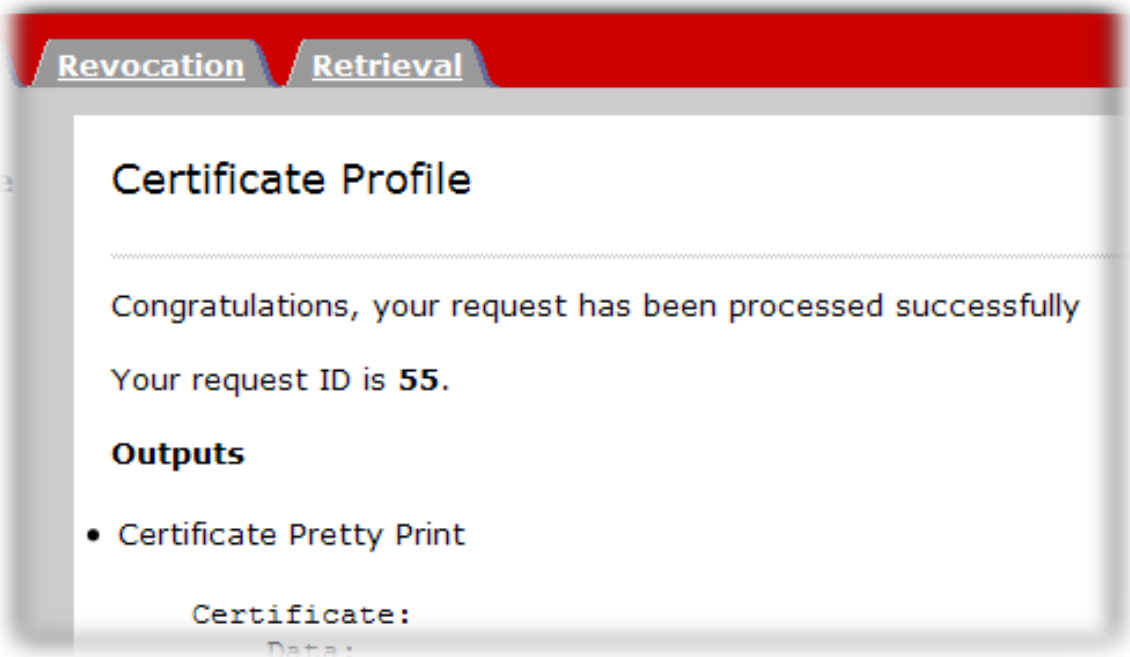
3. 输入要续订的证书的序列号。这可以采用十进制或十六进制格式。



4. 点续订按钮。

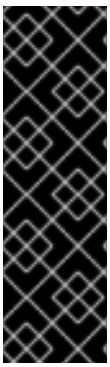
5.

请求已提交。对于基于目录的续订，更新的证书将自动返回。否则，代理将批准续订请求。



5.5.1.1.2. 基于证书的续订

有些用户证书直接存储在浏览器中，因此一些续订表单只需检查浏览器证书数据库来进行续订。如果可以续订证书，则 CA 会自动批准并重新发布证书。



重要

如果被续订的证书 已经 到期，则可能就无法用于基于证书的续订。浏览器客户端可能会禁止任何带有过期证书的 SSL 客户端身份验证。

在这种情况下，必须使用其它续订方法之一续订证书。

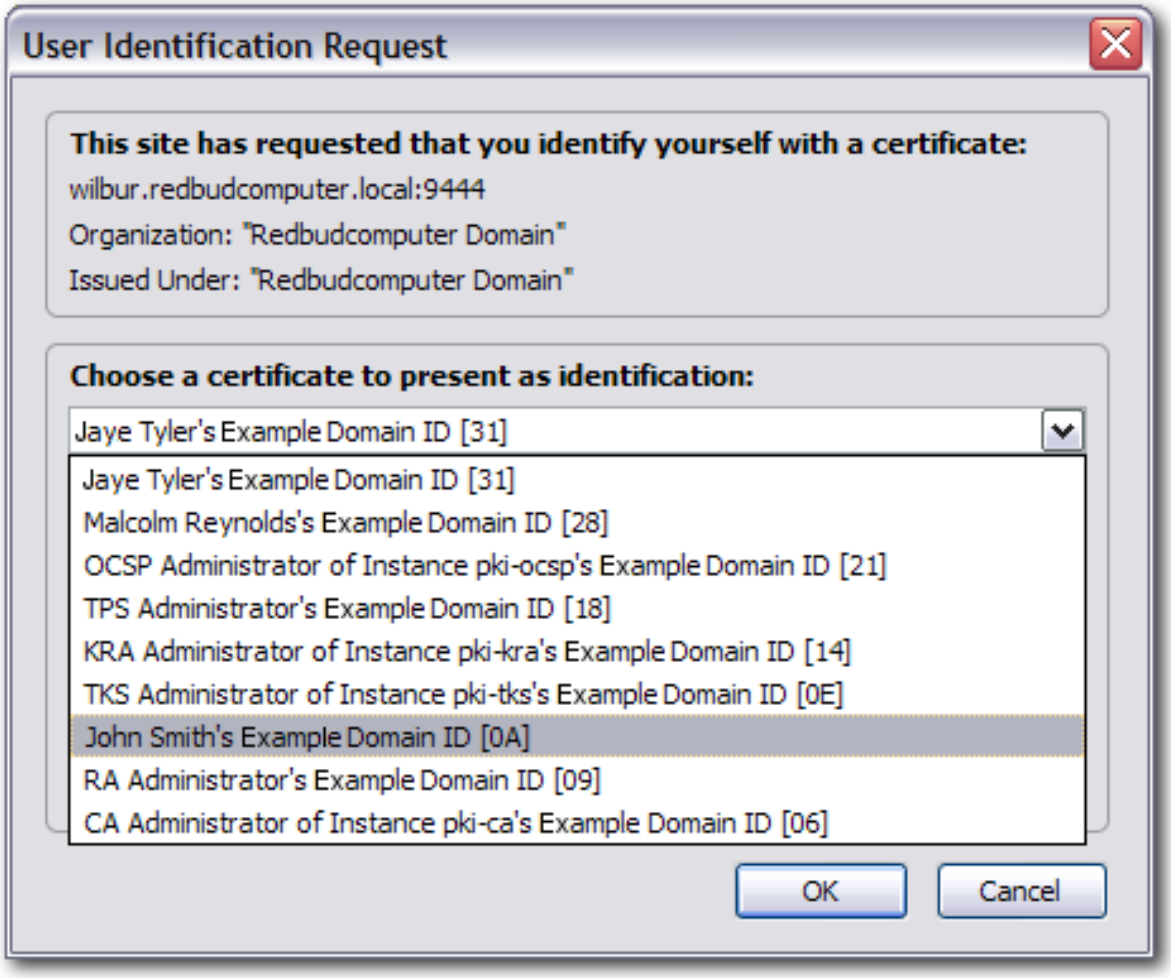
1. 打开发布证书的 CA 的端点服务页面（或其克隆）。

```
https://server.example.com:8443/ca/ee/ca
```

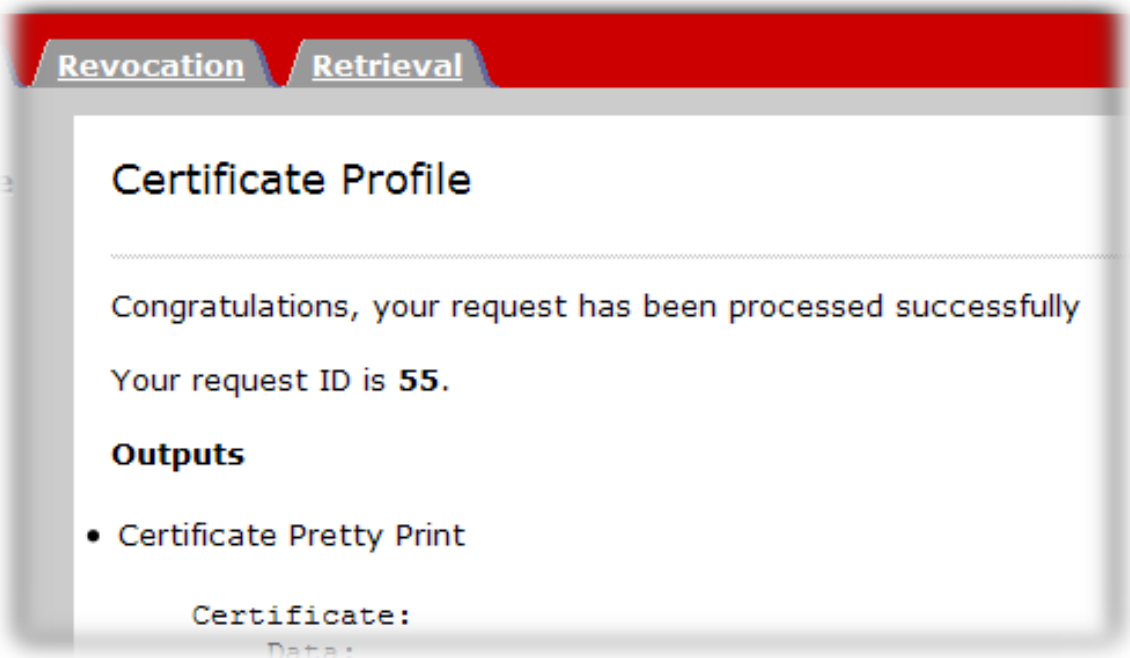
2. 单击要使用的续订表单的名称。

3. *没有输入字段，因此点击 续订 按钮。*

4. *提示时，选择要续订的证书。*



5. *请求将被提交，并且会自动返回更新的证书。*



5.5.1.2. 通过使用相同密钥生成 CSR 续订

有时，原始 CSR 可能不可用。certutil 工具允许一个使用相同键重新生成 CSR，只要该密钥对位于 NSS 数据库中。这可以通过执行以下操作来实现：

1. 在 NSS db 中找到对应的密钥 id：

```
Certutil -d <nssdb dir> -K
```

2. 使用特定密钥生成 CSR：

```
Certutil -d <nssdb dir> -R -k <key id> -s <subject DN> -o <CSR output file>
```

或者，如果一个密钥与 NSS db 中的证书相关联，则可使用 nickname ：

- 使用现有 nickname 生成 CSR：

```
Certutil -d <nssdb dir> -R -k <nickname> -s <subject DN> -o <CSR output file>
```

5.5.2. 通过重新加密证书进行续订

由于重新密钥进行续订基本上是使用与旧证书相同的信息生成新 CSR，只需遵循 第 5.2 节 “创建证

书签名请求” 中描述的任何方法之一。请注意，请输入与旧证书相同的信息。

5.6. 使用 CMC 提交证书请求

这部分论述了使用证书管理 CMS(CMC)注册证书的步骤。

有关使用 CMC 配置和注册证书的一般信息，请参阅：

- 红帽认证系统规划、安装和部署指南中的 CMC 配置部分。
- 红帽认证系统规划、安装和部署指南中的 CMC 注册 部分。
- CMCRequest(1) man page
- CMCResponse(1) man page

CMC 注册方式可以满足您的不同场景的要求。第 5.6.2 节 “CMC 注册过程” 请使用 红帽认证系统规划、安装和部署指南中的 CMC 补充注册 部分，详情。另外，第 5.6.3 节 “实际 CMC 注册方案” 部分可让管理员决定使用哪些机制。

5.6.1. 使用 CMC 注册

CMC 注册允许注册客户端使用 CMCAuth 插件进行身份验证，通过代理证书预签名证书。当收到使用代理证书的有效请求签名时，证书管理器会自动发布证书。

注意

CMC 注册功能被默认启用。除非配置已更改，否则应该不需要启用 CMC 注册身份验证插件或配置集。

CMCAuth 身份验证插件还为客户端提供 CMC 撤销程序。CMC 撤销程序允许客户端具有代理证书签名的证书请求，然后将此类请求发送到证书管理器。当收到使用代理证书签名的有效请求时，证书管理器

会自动撤销证书。CMC 撤销程序可使用 CMCRevoke 命令行工具创建。有关 CMCRevoke 的更多信息，请参阅 第 7.2 节 “执行 CMC Revocation”。

CMC 请求可以通过浏览器终端形式提交，也可以使用 HttpClient 等工具将请求发布到适当的配置集。CMCRequest 工具生成签名证书请求，然后使用 HttpClient 工具或浏览器最终用户表单提交，以自动注册并立即接收证书。

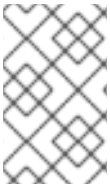
CMCRequest 工具具有简单的命令语法，它通过 .cfg 输入文件中提供的所有配置：

```
CMCRequest /path/to/file.cfg
```

也可以使用 CMCEnroll 工具创建单个 CMC 注册程序，其语法如下：

```
CMCEnroll -d /agent's/certificate/directory -h password -n cert_nickname -r certrequest.file -p
certDB_passwd [-c "comment"]
```

CMCEnroll(1) man page 中详细介绍了这些工具。



注意

在引号中包含空格的值会包括在引号中。

5.6.1.1. 测试 CMCEnroll

1. 使用 certutil 工具创建证书请求。
2. 将 PKCS #10 ASCII 输出复制到文本文件。
3. 运行 CMCEnroll 工具。

例如，如果名为 request34.txt 的输入文件，代理证书存储在浏览器数据库中，则代理证书的证书通用名称是 CertificateManagerAgentsCert，并且证书数据库的密码是 secret，该命令如下：

```
CMCEnroll -d ~jsmith/.mozilla/firefox/1234.jsmith -n "CertificateManagerAgentsCert" -r
/export/requests/request34.txt -p secret
```

- 此命令的输出存储在文件中，该文件与 .out 一起附加到文件名中。

4. 通过端点页面提交签名证书。

   a. 打开"终端"页面。

      https://server.example.com:8443/ca/ee/ca

   b. 从证书配置集列表中选择 CMC 注册表单。

   c. 将输出文件的内容粘贴到此表单 的证书请求 文本区域。

   d. 从粘贴的内容中删除 -----BEGIN NEW CERTIFICATE REQUEST----- 和 ----END NEW CERTIFICATE REQUEST-----。

   e. 填写联系信息并提交表单。

5. 证书会立即被处理并返回。

6. 使用 agent 页面搜索新证书。

5.6.2. CMC 注册过程

使用以下一般流程使用 CMC 请求并发布证书：

1. 使用以下格式创建证书签名请求(CSR)：

   - PKCS #10 格式

- 证书请求消息格式(CRMF)格式

有关使用以下格式创建 CSR 的详情，请参考 第 5.2 节 “创建证书签名请求”。

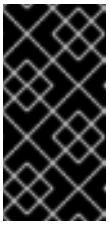
2. 将 admin 证书导入到客户端 NSS 数据库。例如：

- 执行以下命令，从 .p12 文件中提取 admin 客户端证书：

```
$ openssl pkcs12 -in /root/.dogtag/instance/ca_admin_cert.p12 -clcerts -nodes -nokeys -out /root/.dogtag/instance/ca_admin_cert.crt
```

- 根据 红帽认证系统规划、安装和部署指南中的管理证书/密钥加密令牌一节中的指导，验证和导入管理 客户端证书：

```
$ PKICertImport -d . -n "CA Admin - Client Certificate" -t ",," -a -i /root/.dogtag/instance/ca_admin_cert.crt -u C
```



重要

在导入 CA Admin 客户端证书前，请确保已导入所有中间证书和 root CA 证书。

- 导入与证书关联的私钥。

```
$ pki -c password pkcs12-import --pkcs12-file /root/.dogtag/instance/ca_admin_cert.p12 --pkcs12-password-file /root/.dogtag/instance/ca/pkcs12_password.conf
```

3. 为 CMC 请求创建配置文件，如 /home/user_name/cmc-request.cfg，其中包含以下内容：

```
# NSS database directory where CA agent certificate is stored
dbdir=/home/user_name/.dogtag/nssdb/

# NSS database password
password=password

# Token name (default is internal)
```

```
tokenname=internal

# Nickname for signing certificate
nickname=subsystem_admin

# Request format: pkcs10 or crmf
format=pkcs10

# Total number of PKCS10/CRMF requests
numRequests=1

# Path to the PKCS10/CRMF request
# The content must be in Base-64 encoded format.
# Multiple files are supported. They must be separated by space.
input=/home/user_name/file.csr

# Path for the CMC request
output=/home/user_name/cmc-request.bin
```

详情请查看 ***CMCRequest(1) man page。***

4.

创建 ***CMC*** *请求：*

```
$ CMCRequest /home/user_name/cmc-request.cfg
```

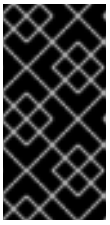
如果命令成功，则 CMCRequest 程序会在请求配置文件的 output 参数中指定的文件中存储 CMC 请求。

5.

为 ***HttpClient*** *创建配置文件，如* ***/home/user_name/cmc-submit.cfg，您要在以后的步骤中使用该文件来向 CA 提交 CMC*** *请求。在创建的文件中添加以下内容：*

```
# PKI server host name
host=server.example.com

# PKI server port number
port=8443

# Use secure connection
secure=true
```

```
# Use client authentication
clientmode=true

# NSS database directory where the CA agent certificate is stored.
dbdir=/home/user_name/.dogtag/nssdb/

# NSS database password
password=password

# Token name (default: internal)
tokenname=internal

# Nickname of signing certificate
nickname=subsystem_admin

# Path for the CMC request
input=/home/user_name/cmc-request.bin

# Path for the CMC response
output=/home/user_name/cmc-response.bin
```



重要

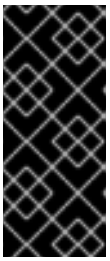
在 nickname 参数中指定的证书的别名必须与之前用于 CMC 请求的证书匹配。

6. **根据您请求的证书类型，将以下参数添加到上一步中创建的配置文件中：**

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=profile_name
```

例如，对于 CA 签名证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCcaCert
```



重要

当代理在下一步中提交 CMC 请求时，这个参数中指定的配置集必须使用 CMCAuth 身份验证插件。在用户初始化的注册中，配置集必须使用 CMCUserSignedAuth 插件。详情请查看 第 9.3 节 “CMC 身份验证插件”。

7. **将 CMC 请求提交到 CA：**

```
$ HttpClient /home/user_name/cmc-submit.cfg
```

-

8. 要将 CMC 响应转换为 PKCS #7 证书链，将 CMC 响应文件传递给 CMCResponse 工具的 -i 参数。例如：

```
$ CMCResponse -i /home/user_name/cmc-response.bin -o /home/user_name/cert_chain.crt
```

5.6.3. 实际 CMC 注册方案

这部分论述了频繁实际的使用场景及其工作流，以便 CA 管理员决定使用哪个 CMC 方法。

有关使用 CMC 注册证书的常规过程，请参阅 第 5.6.2 节 “CMC 注册过程”。

5.6.3.1. 获取系统和服务器证书

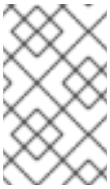
如果 LDAP 或 web 服务器等服务需要 TLS 服务器证书，则此服务器的管理员会根据服务的文档创建一个 CSR，并将其发送到 CA 的代理进行批准。将 第 5.6.2 节 “CMC 注册过程” 中描述的步骤用于此过程。另外，请考虑以下要求：

注册配置集

代理必须使用 第 9.3 节 “CMC 身份验证插件” 中列出的现有 CMC 配置集之一，或者创建一个使用 CMCAuth 身份验证机制的自定义配置集。

CMC 签署证书

对于系统证书，CA 代理必须生成并签署 CMC 请求。为此，请将 CMCRequest 配置文件中的 nickname 参数设置为 CA 代理的 nickname。



注意

CA 代理必须有权访问自己的私钥。

HttpClient TLS Client Nickname

在 CMCRequest 实用程序配置文件中，使用与 HttpClient 配置文件中的 TLS 客户端身份验证相同的证书。

HttpClient servlet Parameter

传递给 **HttpClient** 工具的配置文件中的 **servlet** 是指处理请求的 **CMC servlet** 和注册配置文件。

根据您请求的证书类型，在上一步中创建的配置文件中添加以下条目之一：

- 对于 **CA** 签名证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCcaCert
```

- 对于 **KRA** 传输证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCkraTransportCert
```

- 对于 **OCSP** 签名证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCocspCert
```

- 对于审计签名证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCauditSigningCert
```

- 对于子系统证书：
  - 对于 **RSA** 证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCsubsystemCert
```

  - 对于 **ECC** 证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCECCsubsystemCert
```

- 对于 **TLS** 服务器证书：

- 对于 RSA 证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCserverCert
```

- 对于 ECC 证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCECCserverCert
```

- 对于管理员证书：

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caFullCMCUserCert
```

详情：

- 当代理预签署 CSR 时，由于代理检查 CSR 来进行识别，因此该验证身份被视为已建立的识别。不需要额外的 CMC 特定的身份证。
- PKCS #10 文件已提供 Possession 信息，且不需要额外的 Possession(POP)。
- 在代理预批准的请求中，必须禁用 PopLinkWittnessV2 功能，因为代理会检查身份识别。

5.6.3.2. 获取用户的第一个签名证书

可以通过两种方式批准用户的第一个签名证书：

- 代理为 CMC 请求签名。请参阅 第 5.6.3.2.1 节 “使用代理证书签名 CMC 请求”。
- 证书注册通过共享 Secret 进行身份验证。请参阅 第 5.6.3.2.2 节 “使用共享 Secret 对证书注册进行身份验证”。

5.6.3.2.1. 使用代理证书签名 CMC 请求

使用代理证书签名 CMC 请求的过程与 第 5.6.3.1 节 “获取系统和服务器证书” 中描述的系统和服务

器证书相同。唯一的不同之处在于用户创建 CSR，并将其发送到 CA 代理进行批准。

5.6.3.2.2. 使用共享 Secret 对证书注册进行身份验证

当用户希望获取第一个签名证书时，代理无法批准请求，如 第 5.6.3.2.1 节 “使用代理证书签名 CMC 请求” 所述，您可以使用 Shared Token。使用这个令牌，用户可以获取第一个签名证书。然后，此证书可用于签署用户的其他证书。

在这种情况下，使用 Shared Secret 机制获取用户的第一个签名证书。将以下信息与 第 5.6.2 节 “CMC 注册过程” 一起使用：

1. 以用户或 CA 管理员创建共享令牌。详情请参阅 红帽证书系统规划、安装和部署指南中的 创建共享 Secret 令牌 部分。

   请注意：

   - 如果创建令牌，用户必须将令牌发送到 CA 管理员。
   - 如果 CA 管理员创建了令牌，管理员必须向用户共享用于生成令牌的密码。使用安全的方式传输密码。

2. 作为 CA 管理员，将 Shared Token 添加到 LDAP 中的用户条目中。详情请参阅 Red Hat 证书系统规划、安装和部署指南中的 第 9.4.2.1 节 “将 CMC 共享 Secret 添加到用于证书注册的用户条目” 和 启用 CMC Shared Secret 功能 部分。

3. 使用传递给 CMCRequest 工具的配置文件中的以下参数：

   - identification.enable
   - witness.sharedSecret
   - identityProofV2.enable

- *identityProofV2.hashAlg*
- *identityProofV2.macAlg*
- *request.useSharedSecret*
- *request.privKeyId*

4. *如果 CA 需要，还要使用传递给 CMCRequest 工具的配置文件中的以下参数：*

- *popLinkWitnessV2.enable*
- *popLinkWitnessV2.keyGenAlg*
- *popLinkWitnessV2.macAlg*

5.6.3.3. 为用户获取仅限加密的证书

本节论述了获取只使用加密的证书（使用现有用户签名证书签名的证书）的工作流：

注意

如果用户拥有多个用于不同使用情况的证书（在签名的情况下），用户必须首先获取签名证书。用户拥有签名证书后，就可以使用它进行 概念验证，而无需设置并依赖 CMC 共享 Secret 机制。

有关获取用户的第一个签名证书的详情，请参考 第 5.6.3.2 节 “获取用户的第一个签名证书”。

作为用户：

1.

使用存储在网络安全服务(NSS)数据库中或包含用户签名证书和密钥的智能卡中的加密令牌。

2. 以 PKCS #10 或 CRMF 格式生成 CSR。

注意

如果需要密钥归档，请使用 CRMF 格式。

3. 生成 CMC 请求。

由于这是仅加密的证书，因此私钥无法签名。因此，不包括概念验证(POP)。因此，注册需要两个步骤：如果初始请求成功，则以 EncryptedPOP 控制形式生成 CMC 状态。然后，用户使用响应并生成包含 DecryptedPOP 控制的 CMC 请求，并在第二步提交它。

a. 对于第一步，除了默认参数外，用户还必须在传递到 CMCRequest 工具的配置文件中设置以下参数：

- identification.enable
- witness.sharedSecret
- identityProofV2.enable
- identityProofV2.hashAlg
- identityProofV2.macAlg
- popLinkWitnessV2.enable （CA 需要）
- popLinkWitnessV2.keyGenAlg （CA 需要）

- *popLinkWitnessV2.macAlg （如果需要）*
- *request.privKeyId*

详情请查看 CMCRequest(1) man page。

响应包含：

- *CMC 加密 POP 控制*
- *使用 POP 所需 错误进行 CMCStatusInfoV2 控制*
- *请求 ID*

b. *对于第二个步骤，除了默认参数外，用户还必须在传递到 CMCRequest 工具的配置文件中设置以下参数：*

- *decryptedPop.enable*
- *encryptedPopResponseFile*
- *decryptedPopRequestFile*
- *request.privKeyId*

详情请查看 CMCRequest(1) man page。

5.6.3.3.1. 使用密钥 Archival 的只加密证书示例

要执行密钥存档的注册，生成 CMC 请求，该请求包含在 CRMF 请求中包含该用户的加密私钥。以

下流程假设该用户已经拥有了签名证书。此签名证书的 nickname 在流程中的配置文件中设置。

注意

以下步骤描述了两个条带的提示与加密密钥一起使用，该密钥无法用于签名。如果您使用可签署证书的密钥，请将 **-q POP_SUCCESS** 选项而不是 **-q POP_NONE** 传递给 **single-trip** 的 **CRMFPopClient** 实用程序。

有关将 CRMFPoPClient 与 POP_SUCCESS 搭配使用，请参阅 第 5.2.1.3.1 节 “使用 CRMFPopClient 创建带有密钥 Archival 的 CSR” 和 第 5.2.1.3.2 节 “使用 CRMFPopClient 为基于 SharedSecret 的 CMC 创建 CSR”。

1. 搜索 **KRA** 传输证书。例如：

```
$ pki cert-find --name KRA_transport_certificate_subject_CN
```

2. 使用您在上一步中检索的 **KRA** 传输证书的序列号，将证书存储在文件中。例如，要在 **/home/user_name/kra.cert** 文件中存储带有 **12345** 序列号的证书：

```
$ pki cert-show 12345 --output /home/user_name/kra.cert
```

3. 使用 **CRMFPopClient** 实用程序：

- 使用密钥归档创建 **CSR**：

1. 进入请求证书的用户或实体的证书数据库目录，例如：

```
$ cd /home/user_name/
```

2. 使用 **CRMFPopClient** 实用程序创建 **CRMF** 请求，其中 **RSA** 私钥由 **KRA** 传输证书包装。例如，将请求存储在 **/home/user_name/crmf.req** 文件中：

```
$ CRMFPopClient -d . -p token_password -n subject_DN -q POP_NONE \
  -b /home/user_name/kra.cert -w "AES/CBC/PKCS5Padding" \
  -v -o /home/user_name/crmf.req
```

记录命令显示的私钥的 ID。在第二个行的配置文件中，需要后续步骤中的 **request.privKeyId** 参数作为值。

4. 为 **CRMRequest** 实用程序创建一个配置文件，如 **/home/user_name/cmc.cfg**，其中包含以下内容：

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
input=/home/user_name/crmf.req

#output: full path for the CMC request in binary format
output=/home/user_name/cmc.req

#tokenname: name of token where agent signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for user certificate which will be used
#to sign the CMC full request.
nickname=signing_certificate

#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert8.db which stores the agent certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf
```

5. 创建 **CMC** 请求：

```
$ CMCRequest /home/user_name/cmc.cfg
```

如果命令成功，则 **CMCRequest** 程序会在请求配置文件的 **output** 参数中指定的文件中存储 **CMC** 请求。

6. 为 **HttpClient** 创建配置文件，如 **/home/user_name/cmc-submit.cfg**，您要在以后的步骤中使用该文件来向 **CA** 提交 **CMC** 请求。在创建的文件中添加以下内容：

```
#host: host name for the http server
host=server.example.com
```

```
#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in
#binary format
input=/home/user_name/cmc.req

#output: full path for the response in binary format
output=/home/user_name/cmc-response_round_1.bin

#tokenname: name of token where TLS client authentication cert can be found
#(default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/

#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert8.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=signing_certificate

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitUserSignedCMCFull?profileId=caFullCMCUserSignedCert
```

7. ***将 CMC 请求提交到 CA：***

```
$ HttpClient /home/user_name/cmc-submit.cfg
```

如果命令成功，HTTPClient 程序会在配置文件的 output 参数中指定的文件中存储 CMC 响应。

8. ***通过将响应文件传递给 CMCResponse 实用程序来验证响应。例如：***

```
$ CMCResponse -d /home/user_name/.dogtag/nssdb/ -i /home/user_name/cmc-
response_round_1.bin
```

如果第一个条带成功，CMCResponse 会显示类似如下的输出：

```
Certificates:
  Certificate:
    Data:
      Version:  v3
      Serial Number: 0x1
      Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Issuer: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
      Validity:
        Not Before: Wednesday, May 17, 2017 6:06:50 PM PDT America/Los_Angeles
        Not  After: Sunday, May 17, 2037 6:06:50 PM PDT America/Los_Angeles
      Subject: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
...
Number of controls is 3
Control #0: CMC encrypted POP
  OID: {1 3 6 1 5 5 7 7 9}
   encryptedPOP decoded
Control #1: CMCStatusInfoV2
  OID: {1 3 6 1 5 5 7 7 25}
  BodyList: 1
  OtherInfo type: FAIL
   failInfo=POP required
Control #2: CMC ResponseInfo
  requestID: 15
```

9.

对于第二个条带，为 Decrypt edPOP 创建配置文件，如 /home/user_name/cmc_DecryptedPOP.cfg，您要在以后的步骤中使用。在创建的文件中添加以下内容：

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
#this field is actually unused in 2nd trip
input=/home/user_name/crmf.req

#output: full path for the CMC request in binary format
#this field is actually unused in 2nd trip
output=/home/user_name/cmc2.req

#tokenname: name of token where agent signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
nickname=signing_certificate
```

```
#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert8.db which stores the agent
#certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf

decryptedPop.enable=true
encryptedPopResponseFile=/home/user_name/cmc-response_round_1.bin
request.privKeyId=-25aa0a8aad395ebac7e6a19c364f0dcb5350cfef
decryptedPopRequestFile=/home/user_name/cmc.DecryptedPOP.req
```

10. 创建 **DecryptPOP CMC** 请求：

```
$ CMCRequest /home/user_name/cmc.DecryptedPOP.cfg
```

如果命令成功，**CM CRequest** 程序会在请求配置文件的 **decryptedPopRequestFile** 参数中指定的文件中存储 **CMC** 请求。

11. 为 **HttpClient** 创建配置文件，如 **/home/user_name/decrypted_POP_cmc-submit.cfg**，您可以使用该文件来向 **CA** 提交 **DecryptedPOP CMC** 请求。在创建的文件中添加以下内容：

```
#host: host name for the http server
host=server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in binary format
input=/home/user_name/cmc.DecryptedPOP.req

#output: full path for the response in binary format
output=/home/user_name/cmc-response_round_2.bin

#tokenname: name of token where TLS client authentication cert can be found (default is
internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/
```

```
#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert8.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=singing_certificate

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitUserSignedCMCFull?profileId=caFullCMCUserCert
```

12. **将 DecryptedPOP CMC 请求提交到 CA：**

```
$ HttpClient /home/user_name/decrypted_POP_cmc-submit.cfg
```

如果命令成功，HTTPClient 程序会在配置文件的 output 参数中指定的文件中存储 CMC 响应。

13. **要将 CMC 响应转换为 PKCS #7 证书链，将 CMC 响应文件传递给 CMCResponse 工具的 -i 参数。例如：**

```
$ CMCResponse -i /home/user_name/cmc-response_round_2.bin -o
/home/user_name/certs.p7
```

另外，要以 PEM 格式显示各个证书，请将 -v 传递给 实用程序。

如果第二个条带成功，则 CMCResponse 会显示类似如下的输出：

```
Certificates:
  Certificate:
    Data:
      Version:  v3
      Serial Number: 0x2D
      Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Issuer: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
      Validity:
        Not Before: Thursday, June 15, 2017 3:43:45 PM PDT America/Los_Angeles
        Not  After: Tuesday, December 12, 2017 3:43:45 PM PST America/Los_Angeles
      Subject: CN=user_name,UID=example,OU=keyArchivalExample
```

```
...
Number of controls is 1
Control #0: CMCStatusInfo
  OID: {1 3 6 1 5 5 7 7 1}
  BodyList: 1
  Status: SUCCESS
```

5.7. 执行大量问题

当管理员需要同时提交并生成大量证书时，可以有实例。与 CertificateCertificate Systemnbsp 提供的工具组合；系统可用于发布包含 CA 的证书请求的文件。这个示例步骤使用 PKCS10Client 命令生成 requests 和 sslget 命令，将请求发送到 CA。

1. 由于已编写此过程，需要设置多个变量来识别 CA（主机、端口）以及用于身份验证的项目（代理证书和证书数据库和密码）。例如，通过在终端中导出会话设置这些变量：

```
export d=/var/tmp/testDir
export p=password
export f=/var/tmp/server.csr.txt
export nick="CA agent cert"
export cahost=1.2.3.4
export caport=8443
```

注意

本地系统必须具有一个有效的安全数据库，其中包含代理的证书。设置数据库：

a. **从浏览器中导出或下载代理用户证书和密钥，并保存到文件，如 agent.p12。**

b. **如有必要，为安全数据库创建一个新目录。**

```
mkdir ${d}
```

c. **如有必要，创建新的安全数据库。**

```
certutil -N -d ${d}
```

d. **停止证书证书 Systemnbsp;System 实例。**

```
systemctl stop pki-tomcatd@instance_name.service
```

e. **使用 pk12util 导入证书。**

```
# pk12util -i /tmp/agent.p12 -d ${d} -W p12filepassword
```

如果这个过程成功，命令会输出以下输出：

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

f. **启动证书证书系统nbsp;System 实例。**

```
systemctl start pki-tomcatd@instance_name.service
```

2. **必须设置两个额外的变量。标识要用于处理请求的 CA 配置集的变量，以及用于发送 post 语句的变量，以提供配置集表单的信息。**

```
export
post="cert_request_type=pkcs10&xmlOutput=true&profileId=caAgentServerCert&cert
_request="
export url="/ca/ee/ca/profileSubmitSSLClient"
```

注意

本例将证书请求提交到 caAgentServerCert 配置集（在 post 语句的 profileId 元素中标识）。可以使用任何证书配置集，包括自定义配置集。

3. 测试变量配置。

```
echo ${d} ${p} ${f} ${nick} ${cahost} ${caport} ${post} ${url}
```

4. 使用 （本例）Phy 10Client 生成证书 请求：

```
time for i in {1..10}; do /usr/bin/PKCS10Client -d ${d} -p ${p} -o ${f}.${i} -s
"cn=testms${i}.example.com"; cat ${f}.${i} >> ${f}; done

perl -pi -e 's/\r\n//;s/\+/%2B/g;s/\//%2F/g' ${f}

wc -l ${f}
```

5. 检查 CA 的状态和事务日志。

```
/etc/init.d/pki-ca status

tail -f /var/log/pki-ca/transactions&
```

6. 使用 sslget 将上一步中创建的批量证书请求文件提交到 CA 配置集接口。4例如：

```
cat ${f} | while read thisreq; do /usr/bin/sslget -n "${nick}" -p ${p} -d ${d} -e
${post}${thisreq} -v -r ${url} ${cahost}:${caport}; done
```

5.8. 在 CISCO ROUTER 上注册证书

简单的证书注册协议(SCEP)由 Cisco 设计，是路由器与证书授权机构（如 CA）进行通信的方法，为路由器注册证书。

通常，路由器安装程序在路由器中输入 CA 的 URL 和质询密码（也称为一次性 PIN），并发出命令来启动注册。然后，路由器通过 SCEP 与 CA 通信来生成、请求并检索证书。路由器也可以使用 SCEP 检查待处理请求的状态。

5.8.1. 启用 SCEP 注册

出于安全考虑，CA 中默认禁用 SCEP 注册。要允许路由器注册，必须为 CA 手动启用 SCEP 注册。

1. 停止 CA 服务器，以便您可以编辑配置文件。

```
systemctl stop pki-tomcatd@instance_name.service
```

2. 打开 CA 的 CS.cfg 文件。

```
vim /var/lib/pki/instance_name/ca/conf/CS.cfg
```

3. 将 ca.scep.enable 设置为 true。如果参数不存在，则使用 参数添加一行。

```
ca.scep.enable=true
```

4. 重启 CA 服务器。

```
systemctl start pki-tomcatd@instance_name.service
```

5.8.2. 配置 SCEP 安全设置

管理员可通过几个不同的参数来设置 SCEP 连接的特定安全要求，如不使用相同的证书注册和常规证书注册，或者设置允许的加密算法以防止降级连接强度。这些参数在 表 5.2 “SCEP 安全性的配置参数” 中列出。

表 5.2. SCEP 安全性的配置参数

参数	描述
ca.scep.encryptionAlgorithm	设置默认或首选的加密算法。
ca.scep.allowedEncryptionAlgorithms	设定以逗号分隔的加密算法列表。
ca.scep.hashAlgorithm	设置默认值或首选的哈希算法。
ca.scep.allowedHashAlgorithms	设定以逗号分隔的允许哈希算法列表。
ca.scep.nickname	为 SCEP 通信提供证书的别名。除非设置了此参数，否则默认值是使用 CA 的密钥对和证书。
ca.scep.nonceSizeLimit	设定 SCEP 请求的最大值（以字节为单位）。默认值为 16 字节。

为 SCEP 注册设置安全设置：

1. 停止 CA 服务器，以便您可以编辑配置文件。

```
systemctl stop pki-tomcatd@instance_name.service
```

2. 打开 CA 的 CS.cfg 文件。

```
vim /var/lib/pki/instance_name/ca/conf/CS.cfg
```

3. 设置所需的安全参数，如 表 5.2 “SCEP 安全性的配置参数” 中列出的。如果参数不存在，则将其添加到 CS.cfg 文件中。

```
ca.scep.encryptionAlgorithm=DES3
ca.scep.allowedEncryptionAlgorithms=DES3
ca.scep.hashAlgorithm=SHA1
ca.scep.allowedHashAlgorithms=SHA1,SHA256,SHA512
ca.scep.nickname=Server-Cert
ca.scep.nonceSizeLimit=20
```

4. 重启 CA 服务器。

```
systemctl start pki-tomcatd@instance_name.service
```

5.8.3. 配置路由器进行服务注册

注意

并非所有路由器 IOS 版本都具有相关的加密功能。确保固件镜像具有认证认证机构的互操作性功能。CertificateCertificate Systemnbsp;System SCEP 支持在运行 IOS C2600 Software(C2600-JK9S-M)、版本 12.2(40)、RELEASE SOFTWARE(fc1)的 Cisco 2611 路由器上测试。

在路由器上注册 SCEP 证书之前，请确保正确配置了路由器：

- 路由器必须配置有 IP 地址、DNS 服务器和路由信息。
- 路由器的日期/时间必须正确。
- 必须配置路由器的主机名和 dnsname。

有关配置路由器硬件的说明，请参阅路由器文档。

5.8.4. 为路由器生成 SCEP 证书

以下过程详细介绍了如何为路由器生成 SCEP 证书。

1. 选择随机 PIN。
2. 将 PIN 和路由器的 ID 添加到 flatfile.txt 文件中，以便路由器可以直接与 CA 进行身份验证。例如：

```
vim /var/lib/pki/instance_name/ca/conf/flatfile.txt

UID:172.16.24.238
PWD:Uojs93wkfd0IS
```

务必在 PWD 行后插入空行。

路由器的 IP 地址可以是 IPv4 地址或 IPv6 地址。

第 9.2.4 节 “配置平面文件身份验证” 中描述了使用无格式文件身份验证。

3. 登录到路由器的控制台。在本例中，路由器的名称为 scep ：

```
scep>
```

4. 启用特权命令。

```
scep> enable
```

5. 进入配置模式。

```
scep# conf t
```

6. 从 root 用户开始，为证书链中每个 CA 导入 CA 证书。例如，以下命令序列将链中两个 CA 证书导入到路由器中：

```
scep(config)# crypto ca trusted-root1
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 1
scep(config)# crypto ca trusted-root0
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 0
```

7. 设置 CA 身份，并输入用于访问 SCEP 注册程序的 URL。例如，对于 CA：

```
scep(config)# crypto ca identity CA
scep(ca-identity)# enrollment url http://server.example.com:8080/ca/cgi-bin
scep(ca-identity)# crl optional
```

8. ***获取 CA 的证书。***

```
scep(config)# crypto ca authenticate CA
Certificate has the following attributes:
Fingerprint: 145E3825 31998BA7 F001EA9A B4001F57
% Do you accept this certificate? [yes/no]: yes
```

9. ***生成 RSA 密钥对。***

```
scep(config)# crypto key generate rsa
The name for the keys will be: scep.server.example.com
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]:
Generating RSA keys ...
[OK]
```

10. ***最后，在路由器上生成证书。***

```
scep(config)# crypto ca enroll CA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
password to the CA Administrator in order to revoke your certificate.
For security reasons your password will not be saved in the configuration.
Please make a note of it.

Password: secret
Re-enter password: secret

% The subject name in the certificate will be: scep.server.example.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 57DE391C
% Include an IP address in the subject name? [yes/no]: yes
% Interface: Ethernet0/0
% Request certificate from CA? [yes/no]: yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

% Fingerprint:D89DB555 E64CC2F7 123725B4 3DBDF263

Jan 12 13:41:17.348: %CRYPTO-6-CERTRET: Certificate received from Certificate
```

11. 关闭配置模式。

```
scep(config)# exit
```

12. 为确保路由器已正确注册，请列出存储在路由器中的所有证书。

```
scep# show crypto ca certificates
Certificate
 Status: Available
 Certificate Serial Number: 0C
 Key Usage: General Purpose
 Issuer:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Subject Name Contains:
 Name: scep.server.example.com
 IP Address: 10.14.1.94
 Serial Number: 57DE391C
 Validity Date:
 start date: 21:42:40 UTC Jan 12 2007
 end date: 21:49:50 UTC Dec 31 2008
 Associated Identity: CA

CA Certificate
 Status: Available
 Certificate Serial Number: 01
 Key Usage: Signature
 Issuer:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Subject:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Validity Date:
 start date: 21:49:50 UTC Jan 11 2007
 end date: 21:49:50 UTC Dec 31 2008
 Associated Identity: CA
```

5.8.5. 使用从属 CA

在路由器可以向 CA 验证前，CA 的证书链中的每个 CA 证书都必须导入到路由器中，从 root 开始。例如，以下命令序列将链中两个 CA 证书导入到路由器中：

```
scep(config)# crypto ca trusted-root1
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 1
```

```
scep(config)# crypto ca trusted-root0
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 0
```

如果 CA 证书没有设置 CRL 分布点扩展，请通过将它设置为 可选 来关闭 CRL 要求：

```
scep(ca-root)# crl optional
```

之后，按照 第 5.8.4 节 “为路由器生成 SCEP 证书” 所述设置 CA 身份。

5.8.6. 重新注册路由器

在使用新证书重新注册路由器前，必须移除现有的配置。

1. **删除（零化）现有密钥。**

```
scep(config)# crypto key zeroize rsa
% Keys to be removed are named scep.server.example.com.
Do you really want to remove these keys? [yes/no]: yes
```

2. **删除 CA 身份。**

```
scep(config)# no crypto ca identity CA
% Removing an identity will destroy all certificates received from
the related Certificate Authority.

Are you sure you want to do this? [yes/no]: yes
% Be sure to ask the CA administrator to revoke your certificates.

No enrollment sessions are currently active.
```

5.8.7. 启用调试

路由器在 SCEP 操作期间通过启用 debug 语句提供额外的调试。

```
scep# debug crypto pki callbacks
Crypto PKI callbacks debugging is on

scep# debug crypto pki messages
```

```
Crypto PKI Msg debugging is on

scep# debug crypto pki transactions
Crypto PKI Trans debugging is on

scep#debug crypto verbose
verbose debug output debugging is on
```

5.8.8. 使用 SCEP 发出 ECC 证书

默认情况下，ECC CA 不支持 SCEP out。但是，可以通过使用指定的 RSA 证书来处理以下两个区域的每个区域来临时解决这个问题：

- 加密/解密证书 - 指定具有加密/解密能力的 RSA 证书；（以下示例中的scepRSAcert）
- 签名证书 - 获取要在客户端使用 RSA 证书以签名目的，而不是自签名证书。（以下示例中的认证证书）

例如，如果 scepRSAcert cert 是加密/解密证书，其签名证书是签名证书：

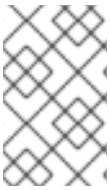
```
sscep enroll -c ca.crt -e scepRSAcert.crt -k local.key -r local.csr -K sign.key -O sign.crt -E 3des -S
sha256 -l cert.crt -u 'http://example.example.com:8080/ca/cgi-bin/pkiclient.exe'
```

第 6 章 使用并配置令牌管理系统：TPS 和 TKS

本章提供了使用硬件安全模块（也称为 HSM 或 令牌 ）的步骤，以生成和存储证书证书证书证书证书证书和密钥。

本章仅包含管理过程。有关令牌管理系统后概念的常规信息，请查看 Red Hat 证书系统 9 规划、安装和部署指南。

6.1. TPS 配置集



注意

有关一般信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 TPS 配置集 部分。

与 CA 注册配置文件（在独立文件或 LDAP 中定义）不同，TPS 配置文件（也称为令牌类型）在 TPS 配置文件 CS.cfg 中定义。

TPS 配置集（令牌类型）配置参数以以下格式设置：

```
op.<explicit op>.<profile id>.<implicit op>.<key type>.*
```

在上面的 中，<explicit op> 和 & lt;implicit op > 是下面 TPS Operations 部分讨论的显式和隐式操作之一，&lt ;key type& gt; 是为每个证书类型给出的名称。

配置参数示例可能类似以下示例：

```
op.enroll.userKey.keyGen.encryption.*
```

6.2. TPS 操作

显式操作

显式操作 是用户调用的操作。显式操作包括 注册 (op.enroll.*)、格式化 (op.format.*)和 pinReset (op.pinReset.*)。

隐式操作

一个 隐式 操作是操作，当处理显式操作时，会发生因为令牌的策略或状态而发生的操作。隐式操作包括 keyGen (op.enroll.userKey.keyGen.*)、续订 (op.enroll.userKey.renewal.*)、update.applet (op.enroll.userKey.update.applet.*)和密钥更新(op.enroll.userKey.update.symmetrics.*)。

一些隐式操作按密钥类型进行控制。这包括 恢复、 serverKeygen 和 revocation 。

以下 TPS 配置集示例指定要在服务器端生成的用户密钥：

```
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true
```

另外，以下示例告知 TPS，其密钥被入侵的令牌应在状态过渡过程中撤销撤销原因 1 的撤销证书：

```
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert.reason=1
```

根据 RFC 5280，可以撤销的原因及其代码，如下所示：

表 6.1. 撤销原因和代码

原因	代码
Unspecified	0
keyCompromise	1
CACompromise	2
affiliationChanged	3
取代	4
cessationOfOperation	5
certificateHold	6
removeFromCRL	8
privilegeWithdrawn	9

原因	代码
AACompromise	10

6.3. 令牌策略

本节提供了可基于 TPS UI 针对每个令牌应用的令牌策略列表。Ech 部分将显示每个策略如何反映在配置中。

注意

有关一般信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 令牌策略 部分。

策略是每个策略的集合，每个策略由分号(";"")分开。每个策略都可以打开或关闭关键字 YES 或 NO。以下列表中的每个策略都会在其默认值中引入 - 如果策略字符串中不存在该设置，则由 TPS 执行的操作。

RE_ENROLL=YES

此策略控制令牌是否允许重新注册操作。这允许重新注册的令牌（带有证书）并给出新的令牌。如果设置为 NO，则服务器在尝试重新注册时会返回错误。

这个策略不需要特殊配置。注册将继续进行标准注册配置文件，该配置文件可能会最初注册令牌。

RENEW=NO;RENEW_KEEP_OLD_ENC_CERTS=YES

续订可让令牌在其配置集生成的证书放置在令牌上。如果 RENEW 设为 YES，则从企业安全客户端(ESC)的简单注册将会导致续订，而不是如上所示的重新注册。

RENEW_KEEP_OLD_ENC_CERTS 设置决定续订操作是否保留加密证书的旧版本。保留前面的证书可让用户访问使用旧证书加密的数据。将此选项设置为 NO，表示任何使用旧证书加密的任何内容都将不再可以被恢复。

配置：

```
op.enroll.userKey.renewal.encryption.ca.conn=ca1
op.enroll.userKey.renewal.encryption.ca.profileId=caTokenUserEncryptionKeyRenewal
```

```
op.enroll.userKey.renewal.encryption.certAttrId=c2
op.enroll.userKey.renewal.encryption.certId=C2
op.enroll.userKey.renewal.encryption.enable=true
op.enroll.userKey.renewal.encryption.gracePeriod.after=30
op.enroll.userKey.renewal.encryption.gracePeriod.before=30
op.enroll.userKey.renewal.encryption.gracePeriod.enable=false
op.enroll.userKey.renewal.keyType.num=2
op.enroll.userKey.renewal.keyType.value.0=signing
op.enroll.userKey.renewal.keyType.value.1=encryption
op.enroll.userKey.renewal.signing.ca.conn=ca1
op.enroll.userKey.renewal.signing.ca.profileId=caTokenUserSigningKeyRenewal
op.enroll.userKey.renewal.signing.certAttrId=c1
op.enroll.userKey.renewal.signing.certId=C1
op.enroll.userKey.renewal.signing.enable=true
op.enroll.userKey.renewal.signing.gracePeriod.after=30
op.enroll.userKey.renewal.signing.gracePeriod.before=30
op.enroll.userKey.renewal.signing.gracePeriod.enable=false
```

这种类型的续订配置会对基本 用户Key 标准注册配置集进行镜像，并带有几个特定于续订的设置。需要这种奇偶校验的原因是，我们需要续订最初注册到令牌中的 certs 数量和类型，然后再进行续订。

FORCE_FORMAT=NO

如果启用，这个策略会导致每个注册操作都提示操作。这是一个最后一个步骤选项，允许在不用户将其返回到管理员的情况下重置令牌。如果设置为 YES，则用户启动的每个注册操作将导致格式发生，可能会将令牌重置为已格式化状态。

不需要额外的配置。为执行标准格式操作相同的 TPS 配置集出现简单的格式。

PIN_RESET=NO

此策略决定了已经注册的令牌是否可以使用 ESC 执行显式"pin reset"更改。此值必须设置为 YES，否则尝试的操作将拒绝服务器出错。

配置：

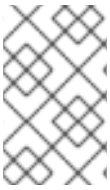
```
op.enroll.userKey.pinReset.enable=true
op.enroll.userKey.pinReset.pin.maxLen=10
op.enroll.userKey.pinReset.pin.maxRetries=127
op.enroll.userKey.pinReset.pin.minLen=4
```

在上例中，minLen 和 maxLen 的设置会限制选择密码的长度，maxRetries 设置将令牌设置为仅在锁定前重试次数。

可使用最新的 TPS 用户界面轻松编辑 TPS 策略。导航到需要更改策略的令牌，并点击 Edit。这将弹出一个对话框，供您编辑字段，该字段是连续运行半以冒号分隔的策略的集合。每个支持的策略都必须设置为 < POLICYNAME>=YES 或 <POLICYNAME> =NO，才能被 TPS 识别。

6.4. 令牌操作和策略处理

本节讨论涉及令牌的主要操作（显式和隐式）。以下列表将讨论每个功能及其配置。



注意

有关一般信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 令牌策略部分。

格式

Format 操作（用户发起）以完全空白状态获取令牌，由制造商提供，并在其上加载 Coolkey 小程序。

配置示例：

```
#specify that we want authentication for format. We almost always want this at true:
op.format.userKey.auth.enable=true
#specify the ldap authentication configuration, so TPS knows where to validate
credentials:
op.format.userKey.auth.id=ldap1
#specify the connection the the CA
op.format.userKey.ca.conn=ca1
#specify id of the card manager applet on given token
op.format.userKey.cardmgr_instance=A0000000030000

#specify if we need to match the visa cuid to the nist sp800sp derivation algorithm KDD
value. Mostly will be false:
op.format.userKey.cuidMustMatchKDD=false

#enable ability to restrict key changoever to a specific range of key set:
op.format.userKey.enableBoundedGPKeyVersion=true
#enable the phone home url to write to the token:
op.format.userKey.issuerinfo.enable=true
#actual home url to write to token:
op.format.userKey.issuerinfo.value=http://server.example.com:8080/tps/phoneHome
#specify whether to request a login from the client. Mostly true, external reg may want this
to be false:
op.format.userKey.loginRequest.enable=true
#Actual range of desired keyset numbers:
op.format.userKey.maximumGPKeyVersion=FF
op.format.userKey.minimumGPKeyVersion=01
```

```
#Whether or not to revoke certs on the token after a format, and what the reason will be if so:
op.format.userKey.revokeCert=true
op.format.userKey.revokeCert.reason=0
#This will roll back the reflected keyyset version of the token in the tokendb. After a failed key changeover operation. This is to keep the value in sync with reality in the tokendb. Always false, since this version of TPS avoids this situation now:
op.format.userKey.rollbackKeyVersionOnPutKeyFailure=false

#specify connection to the TKS:
op.format.userKey.tks.conn=tks1
#where to get the actual applet file to write to the token:
op.format.userKey.update.applet.directory=/usr/share/pki/tps/applets
#Allows a completely blank token to be recognized by TPS. Mostly should be true:
op.format.userKey.update.applet.emptyToken.enable=true
#Always should be true, not supported:
op.format.userKey.update.applet.encryption=true
#Actual version of the applet file we want to upgrade to. This file will have a name something like: 1.4.54de7a99.ijc:
op.format.userKey.update.applet.requiredVersion=1.4.54de790f
#Symm key changeover:
op.format.userKey.update.symmetricKeys.enable=false
op.format.userKey.update.symmetricKeys.requiredVersion=1
#Make sure the token db is in sync with reality. Should always be true:
op.format.userKey.validateCardKeyInfoAgainstTokenDB=true
```

注册

基本注册操作采用格式化的令牌，并将 certs 和密钥放在令牌中以个性化令牌。以下配置示例将介绍如何控制它。

示例显示了不处理续订和内部恢复的基本注册。这里未讨论的设置在 Format 部分中提供，或者没有关键。

```
op.enroll.userKey.auth.enable=true
op.enroll.userKey.auth.id=ldap1
op.enroll.userKey.cardmgr_instance=A0000000030000
op.enroll.userKey.cuidMustMatchKDD=false

op.enroll.userKey.enableBoundedGPKeyVersion=true
op.enroll.userKey.issuerinfo.enable=true
op.enroll.userKey.issuerinfo.value=http://server.example.com:8080/tps/phoneHome

#configure the encryption cert and keys  we want on the token:

#connection the the CA, which issues the certs:
op.enroll.userKey.keyGen.encryption.ca.conn=ca1
#Profile id we want the CA to use to issue our encrytion cert:
op.enroll.userKey.keyGen.encryption.ca.profileId=caTokenUserEncryptionKeyEnrollment

#These two cover the indexes of the certs written to the token. Each cert needs a unique index or “slot”. In our sample the enc cert will occupy slot 2 and the signing cert, shown
```

```
later, will occupy slot 1. Avoid overlap with these numbers:
op.enroll.userKey.keyGen.encryption.certAttrId=c2
op.enroll.userKey.keyGen.encryption.certId=C2

op.enroll.userKey.keyGen.encryption.cuid_label=$cuid$
#specify size of generated private key:
op.enroll.userKey.keyGen.encryption.keySize=1024
op.enroll.userKey.keyGen.encryption.keyUsage=0
op.enroll.userKey.keyGen.encryption.keyUser=0
#specify pattern for what the label of the cert will look like when the cert nickname is
displayed in browsers and mail clients:
op.enroll.userKey.keyGen.encryption.label=encryption key for $userid$
#specify if we want to overwrite certs on a re-enrollment operation. This is almost always
the case:
op.enroll.userKey.keyGen.encryption.overwrite=true

#The next several settings specify the capabilities that the private key on the final token
will inherit. For instance this will determine if the cert can be used for encryption or digital
signatures. There are settings for both the private and public key.

op.enroll.userKey.keyGen.encryption.private.keyCapabilities.decrypt=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.derive=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.encrypt=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.private=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.sensitive=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.sign=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.token=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.unwrap=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.verify=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.encryption.privateKeyAttrId=k4
op.enroll.userKey.keyGen.encryption.privateKeyNumber=4
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.derive=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.encrypt=true
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.private=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.sensitive=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.sign=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.token=true
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.verify=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.wrap=true

#The following index numbers correspond to the index or slot that the private and public
keys occupy. The common formula we use is that the public key index will be 2 * cert id + 1,
and the private key index, shown above will be 2 * cert id. In this example the cert id is 2,
so the key ids will be 4 and 5 respectively. When composing these, be careful not to create
conflicts. This applies to the signing key section below.

op.enroll.userKey.keyGen.encryption.publicKeyAttrId=k5
op.enroll.userKey.keyGen.encryption.publicKeyNumber=5
```

```
#specify if, when a certificate is slated for revocation, based on other rules, we want to check to see if some other token is using this cert in a shared situation. If this is set to true, and this situation is found the cert will not be revoked until the last token wants to revoke this cert:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.holdRevocationUntilLastCredential=false

#specify, if we want server side keygen, if we want to have that generated key archived to the drm. This is almost always the case, since we want the ability to later recover a cert and its encryption private key back to a new token:
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
#connection to drm to generate the key for us:
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
#specify server side keygen of the encryption private key. This most often will be desired:
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true

#This setting tells us how many certs we want to enroll for this TPS profile, in the case "userKey". Here we want 2 total certs. The next values then go on to index into the config what two types of certs we want, signing and encryption:
op.enroll.userKey.keyGen.keyType.num=2
op.enroll.userKey.keyGen.keyType.value.0=signing
op.enroll.userKey.keyGen.keyType.value.1=encryption

#configure the signing cert and keys we want on the token the settings for these are similar to the encryption settings already discussed, except the capability flags presented below, since this is a signing key.

op.enroll.userKey.keyGen.signing.ca.conn=ca1
op.enroll.userKey.keyGen.signing.ca.profileId=caTokenUserSigningKeyEnrollment
op.enroll.userKey.keyGen.signing.certAttrId=c1
op.enroll.userKey.keyGen.signing.certId=C1
op.enroll.userKey.keyGen.signing.cuid_label=$cuid$
op.enroll.userKey.keyGen.signing.keySize=1024
op.enroll.userKey.keyGen.signing.keyUsage=0
op.enroll.userKey.keyGen.signing.keyUser=0
op.enroll.userKey.keyGen.signing.label=signing key for $userid$
op.enroll.userKey.keyGen.signing.overwrite=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.derive=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.encrypt=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.private=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.sensitive=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.sign=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.signRecover=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.token=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.verify=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.signing.privateKeyAttrId=k2
op.enroll.userKey.keyGen.signing.privateKeyNumber=2
op.enroll.userKey.keyGen.signing.public.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.derive=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.encrypt=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.private=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.sensitive=false
```

```
op.enroll.userKey.keyGen.signing.public.keyCapabilities.sign=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.token=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.verify=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.verifyRecover=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.signing.publicKeyAttrId=k3
op.enroll.userKey.keyGen.signing.publicKeyNumber=3
```

pin Reset

第 6.3 节 “令牌策略” 中讨论 pin reset 的配置，因为 pin reset 依赖于策略来决定其是否是合法执行的。

续订

第 6.3 节 “令牌策略” 中会讨论续订的配置，因为续订依赖于策略来确定是否必须按照已注册的令牌执行或不受注册的令牌执行。

恢复

当 TPS 用户界面用户将之前的活跃令牌标记为一个不良状态（如"lost"或"destroyed）"时，恢复会被隐式设置。发生这种情况后，下一个注册了同一用户的新令牌后，会遵循以下配置，以将证书从用户旧令牌恢复到这个新令牌。

此操作的最终结果是用户将具有一个新的物理令牌，该令牌可能包含从旧令牌中恢复的加密证书，以便用户可以根据需要继续加密和解密数据。通常还会将新的签名证书放在这个令牌上，如以下示例配置示例所示。

以下是支持状态列表，其中令牌可以手动放在 TPS 用户界面中，如配置中所示：

- *tokendb._069=# - DAMAGED(1): Corresponds 在恢复配置中 销毁。当令牌被物理损坏时使用。*
- *tokendb._070=# - PERM_LOST(2): Corresponds to keyCompromisein recovery configuration.永久丢失令牌时使用。*
- *tokendb._071=# - SUSPENDED(3): Corresponds to onHold in the restore configuration.临时使用令牌时，用户希望再次查找它。*

- *tokendb._072=# - TERMINATED(6): Corresponds 在恢复配置中 终止。用于出于内部原因而取出来自服务的令牌。*

恢复配置示例：

```
#When a token is marked destroyed, don’t revoke the certs on the token unless all other tokens do not have the certs included:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.holdRevocationUntilLastCredential=false
#specify if we even want to revoke certs a token is marked destroyed:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeCert=false
#if we want to revoke any certs here, specify the reason for revocation that will be sent to the CA:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeCert.reason=0
#speficy if we want to revoke expired certs when marking the token destroyed:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeExpiredCerts=false
```

其他设置用于指定在对新令牌进行恢复操作时应使用什么类型的静态恢复（当原始令牌被标记为销毁时）。支持以下方案：

- *恢复 Last(RecoverLast)：恢复要放置在令牌中的最新加密证书。*
- *生成新密钥和 Recover Last(GenerateNewKeyAndRecoverLast): Same as Recover Last，但也会生成新的加密证书并将其上传到令牌。然后新令牌将有两个证书。*
- *生成新密钥(GenerateNewKey): 生成新加密证书并将其放在令牌中。不要恢复任何旧的证书。*

例如：

```
op.enroll.userKey.keyGen.encryption.recovery.destroyed.scheme=RecoverLast
```

以下配置示例确定如何恢复标记为永久丢失的令牌：

```
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.holdRevocationUntilLastCredential=false
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert.reason=1
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeExpiredCerts=false
```

```
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.scheme=GenerateNewKey

# Section when a token is marked terminated.

op.enroll.userKey.keyGen.encryption.recovery.terminated.holdRevocationUntilLastCredent
ial=false
op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeCert.reason=1
op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeExpiredCerts=false
op.enroll.userKey.keyGen.encryption.recovery.terminated.scheme=GenerateNewKey

# This section details the recovery profile with respect to which certs and of what kind get
recovered on the token.

op.enroll.userKey.keyGen.recovery.destroyed.keyType.num=2
op.enroll.userKey.keyGen.recovery.destroyed.keyType.value.0=signing
op.enroll.userKey.keyGen.recovery.destroyed.keyType.value.1=encryption
```

最后，以下示例确定系统将对旧令牌上的签名证书做了哪些操作。在大多数情况下，应使用 GenerateNewKey 恢复方案来避免使用签名私钥的多个副本（例如：在新令牌中恢复的另外一个新令牌），另一个由其他人永久丢失但由其他人发现的旧令牌。

```
op.enroll.userKey.keyGen.recovery.keyCompromise.keyType.value.0=signing
op.enroll.userKey.keyGen.recovery.keyCompromise.keyType.value.1=encryption
op.enroll.userKey.keyGen.recovery.onHold.keyType.num=2
op.enroll.userKey.keyGen.recovery.onHold.keyType.value.0=signing
op.enroll.userKey.keyGen.recovery.onHold.keyType.value.1=encryption

op.enroll.userKey.keyGen.signing.recovery.destroyed.holdRevocationUntilLastCredential=
false
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeCert.reason=0
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.destroyed.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.holdRevocationUntilLastCred
ential=false
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeCert.reason=1
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.onHold.holdRevocationUntilLastCredential=fal
se
op.enroll.userKey.keyGen.signing.recovery.onHold.revokeCert=true

op.enroll.userKey.keyGen.signing.recovery.onHold.revokeCert.reason=6
op.enroll.userKey.keyGen.signing.recovery.onHold.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.onHold.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.terminated.holdRevocationUntilLastCredential
=false
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeCert.reason=1
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.terminated.scheme=GenerateNewKey
```

```
# Configuration for the case when we mark a token “onHold” or temporarily lost

op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.revokeCert=true
op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.revokeCert.reason=0
op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.scheme=RecoverLast
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.num=2
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.value.0=signing
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.value.1=encryption
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.revokeCert=true
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.revokeCert.reason=0
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.scheme=GenerateNewKey
```

小应用程序更新

以下示例演示了如何配置 Coolkey 小程序更新操作。此操作可以在格式、注册和 PIN 重置操作的过程中执行：

```
op.format.userKey.update.applet.directory=/usr/share/pki/tps/applets
op.format.userKey.update.applet.emptyToken.enable=true
op.format.userKey.update.applet.encryption=true
op.format.userKey.update.applet.requiredVersion=1.4.54de790f
```

其中一些选项已在 Format 部分进行了演示。它们提供必要的信息，以确定是否应该允许进行应用程序升级，在哪里查找小应用程序文件，以及将令牌升级到的小版本。requiredVersion 中的版本映射到 目录 下的文件名。

密钥更新

此操作可在格式、注册和 PIN 重置操作的过程中进行，允许用户从制造商提供的默认全局平台设置版本进行升级。

TPS

以下选项指示 TPS 在代表给定令牌请求的下一个格式操作时将密钥从 1 升级到 2。完成后，TKS 必须生成将写入令牌的三个新密钥，之后，令牌必须与同一 TPS 和 TKS 安装一起使用，否则它将被锁定。

```
op.format.userKey.update.symmetricKeys.enable=true
op.format.userKey.update.symmetricKeys.requiredVersion=2
```

您还可以指定小于当前版本来降级密钥集。

TKS

如上所述，必须将 TKS 配置为生成要写入令牌的新密钥。首先，新的 master 键标识符 02 必须映射到 TKS CS.cfg 中的 PKCS #11 对象 nickname，如下例所示：

```
tks.mk_mappings.#02#01=internal:new_master
tks.defKeySet.mk_mappings.#02#01=internal:new_master
```

以上会将一个键设置号映射到 TKS NSS 数据库中存在的实际 master 密钥。

Master 密钥由 ID 标识，如 01。TKS 将这些 ID 映射到映射的 masterKeyId 部分中指定的 PKCS #11 对象 nicknames。因此，第一个数字会在主密钥版本被更新时更新，第二个数字保持一致。

当尝试从 1 升级到版本 2 时，映射决定了如何查找主键别名，这些名称将用于生成新密钥集的 3 个部分。

上例中的 internal 设置引用主密钥所在的令牌的名称。它还可以是带有名称（如 nethsm ）的外部 HSM 模块。strong new_master 是 master 键 nickname 本身的示例。

6.5. 内部注册

注意

有关一般信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 TPS 配置集 部分。

如果是 内部注册，则 TPS 配置集（令牌类型）由 Mapping Resolver 决定。与外部 注册 不同的是，在配置集本身中定义身份验证信息。例如：

```
op.enroll.userKey.auth.enable=true
op.enroll.userKey.auth.id=ldap1
```

外部注册的另一个区别在于，CA 和 KRA 连接器信息是在每个配置集的密钥类型下定义。例如：

```
op.enroll.userKey.keyGen.encryption.ca.conn=ca1
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
```

但是，TKS 连接器信息为每个配置集定义：

```
op.enroll.userKey.tks.conn=tks1
```

注意

在内部和外部注册间切换注册类型意味着您必须格式化所有之前注册的令牌，然后才能继续使用它们。

6.6. 外部注册

外部注册从经过身份验证的用户 LDAP 记录中获取令牌类型（TPS 配置集）。它还允许在同一用户记录中指定证书/密钥恢复信息。

外部注册 TPS 配置集与前面讨论的内部注册配置文件类似。它允许您为客户端和服务器端密钥生成指定新证书注册。与内部注册不同，它允许您选择特定证书（及其匹配密钥）以检索并加载到令牌中。

注意

在内部和外部注册间切换注册类型意味着您必须格式化所有之前注册的令牌，然后才能继续使用它们。

6.6.1. 启用外部注册

外部注册只能为整个 TPS 实例全局启用。以下示例显示了与外部注册相关的一组全局配置参数：

```
externalReg.allowRecoverInvalidCert.enable=true
externalReg.authId=ldap1
externalReg.default.tokenType=externalRegAddToken
externalReg.delegation.enable=true
externalReg.enable=true
externalReg.recover.byKeyID=false
externalReg.format.loginRequest.enable=true
externalReg.mappingResolver=keySetMappingResolver
```

6.6.2. 自定义用户 LDAP 记录属性名称

以下示例中显示了与外部注册相关的身份验证参数（具有其默认值）：

```
auths.instance.ldap1.externalReg.certs.recoverAttributeName=certsToAdd
auths.instance.ldap1.externalReg.cuidAttributeName=tokenCUID
auths.instance.ldap1.externalReg.tokenTypeAttributeName=tokenType
```

LDAP 记录属性名称可以在此处自定义。确保用户的 LDAP 记录的实际属性与此配置匹配。

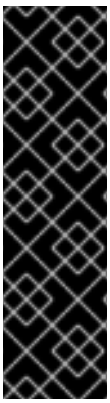
6.6.3. 配置 certsToAdd 属性

certsToAdd 属性采用以下格式的多个值：

```
<cert serial # in decimal>,<CA connector ID>,<key ID>,<kra connector ID>
```

例如：

```
59,ca1,0,kra1
```



重要

默认情况下，密钥恢复会按证书搜索密钥，使 < key ID> 值不相关。但是，可以选择性地将 TPS 配置为使用此属性搜索键，因此通常更易于将值设为 0。该值无效，这可避免获得不匹配的密钥的可能性。

不建议通过密钥 ID 恢复，因为 KRA 无法验证证书是否与这种情况中的密钥匹配。

当只使用证书和 CA 信息指定 certsToAdd 属性时，TPS 会假定问题中的证书已在令牌中已存在，并且应该保留它。这一概念被称为 关键保留。

以下示例显示了用户 LDAP 记录中的相关属性：

```
tokenType: externalRegAddToToken
certstoadd: 59,ca1,0,kra1
certstoadd: 134,ca1,0,kra1
Certstoadd: 24,ca1
```

6.6.4. 用户匹配强制的令牌

另外，您可以设置系统以便用于注册的令牌必须与用户记录唯一 ID(CUID)属性匹配。如果记录中缺少此属性(tokencuid)，则不会强制执行 CUID 匹配。

```
Tokencuid: a10192030405028001c0
```

有关外部注册的另一个属性是绕过每个令牌上的令牌策略。

注意

对于要在外部注册中"接收"的证书和密钥，在用户 LDAP 记录中指定 CA 和 KRA 连接器信息。与证书/密钥相关的 TPS 配置集中指定的 CA 和/或 KRA 连接器信息都会被忽略。

```
certstoadd: 59,ca1,0,kra1
```

6.6.5. 委托支持

委托支持在身份验证（登录、数据加密和解密或签名方面）方面，用户就可代表其代表谁进行委托委托（例如，公司执行者具有一个或多个委托）。

一个示例场景可能是每个委托都有自己的令牌，用于代表执行执行。此令牌包含以下证书和密钥（由 TPS 配置集终止）的组合：

- Authentication certificate/keys：CN 包含委派的名称和唯一 ID。主题备用名称(SAN)扩展包含执行的主名称(UPN)。
- 加密证书：执行主加密证书的确切副本。
- 签名证书：CN 包含委派的名称和唯一 ID。SAN 包含执行执行的 RFC822Name。

使用以下参数来启用委托支持：

```
externalReg.delegation.enable=true
```

重要

要临时解决这个问题，请手动将 op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId 参数设置为 /var/lib/pki/instance_name/tps/conf/CS.cfg 文件来 caTokenUserDelegateAuthKeyEnrollment:

```
op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

6.6.6. SAN 和 DN 模式

auths.instance. <authID > .ldapStringAttributes in authentication instance configuration 指定在身份验证过程中检索哪些属性。例如：

```
auths.instance.ldap1.ldapStringAttributes=mail,cn,uid,edipi,pcc,firstname,lastname,exec-edipi,exec-pcc,exec-mail,certsToAdd,tokenCUID,tokenType
```

从用户的 LDAP 记录中检索后，可以引用这些属性的值，并使用它组成证书的 Subject Alternative Name(SAN)或以 $auth. <attribute name> $.例如：

```
op.enroll.delegateIEtoken.keyGen.authentication.SANpattern=$auth.exec-edipi$.$auth.exec-pcc$@EXAMPLE.com
op.enroll.delegateIEtoken.keyGen.authentication.dnpattern=cn=$auth.firstname$.$auth.lastname$.$auth.edipi$,e=$auth.mail$,o=TMS Org
```

当在 SAN 和 DN 的 TPS 配置集中使用模式时，务必要确保正确设置 TPS 配置集中指定的 CA 注册配置集。例如：

在 TPS 上，在配置集 delegateIEtoken 中

```
op.enroll.delegateIEtoken.keyGen.authentication.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

在 CA 中，在注册配置集 caTokenUserDelegateAuthKeyEnrollment 中

subjectDNInputImpl 插件必须指定为输入之一，以便上面的 TPS 配置集指定 DN：

```
input.i2.class_id=subjectDNInputImpl
input.i2.name=subjectDNInputImpl
```

同样，要允许上述 TPS 配置集指定 SAN，必须指定 subjectAltNameExtInputImpl 插件：

```
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl
```

还必须指定 subjAltExtpattern：

```
policyset.set1.p6.default.params.subjAltExtPattern_0=
(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.req_san_pattern_0$
```

在上例中，OID 1.3.6.1.4.1.311.20.2.3 是 User Principal Name(UPN)的 OID，而 request.req_san_pattern_0 是 delegateIEtoken SAN 模式中指定的第一个 SAN 模式。

您可以同时指定多个 SAN。在 TPS 端，在 SANpattern 中指定多个 SAN，用逗号(",")分隔。在 CA 端，需要以以下格式定义对应的 subjAltExtPattern ：

```
policyset.<policy set id>.<policy id>.default.params.subjAltExtPattern_<ordered number>=
```

在上面的 中，& lt;ordered number > 以 0 开头，并增加 TPS 端指定的每个 SAN 模式：

```
policyset.set1.p6.default.params.subjAltExtPattern_0=
policyset.set1.p6.default.params.subjAltExtPattern_1=
...
```

以下是一个完整的示例：

例 6.1. SANpattern 和 DNpattern 配置

LDAP 记录包含以下信息：

```
givenName: user1a
mail: user1a@example.org
firstname: user1a
edipi: 123456789
pcc: AA
exec-edipi: 999999999
exec-pcc: BB
exec-mail: user1b@EXAMPLE.com
tokenType: delegateISEtoken
certstoadd: 59,ca1,0,kra1
```

TPS External Registration 配置集 委派IEtoken 包含：

- *SANpattern:*

```
op.enroll.delegateISEtoken.keyGen.authentication.SANpattern=$auth.exec-
edipi$.$auth.exec-pcc$@EXAMPLE.com
```

- *DNPattern ：*

```
op.enroll.delegateISEtoken.keyGen.authentication.dnpattern=cn=$auth.firstname$.
$auth.lastname$.$auth.edipi$,e=$auth.mail$,o=TMS Org
```

CA caTokenUserDelegateAuthKeyEnrollment contains:

```
input.i2.class_id=subjectDNInputImpl
input.i2.name=subjectDNInputImpl
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl

policyset.set1.p6.constraint.class_id=noConstraintImpl
policyset.set1.p6.constraint.name=No Constraint
policyset.set1.p6.default.class_id=subjectAltNameExtDefaultImpl
policyset.set1.p6.default.name=Subject Alternative Name Extension Default
policyset.set1.p6.default.params.subjAltExtGNEnable_0=true
policyset.set1.p6.default.params.subjAltExtPattern_0=
(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.req_san_pattern_0$
policyset.set1.p6.default.params.subjAltExtType_0=OtherName
policyset.set1.p6.default.params.subjAltNameExtCritical=false
policyset.set1.p6.default.params.subjAltNameNumGNs=1
```

然后，生成的证书包含：

```
Subject: CN=user1a..123456789,E=user1a@example.org,O=TMS Org
Identifier: Subject Alternative Name - 2.5.29.17
Critical: no
Value:
 OtherName: (UTF8String)1.3.6.1.4.1.311.20.2.3,999999999.BB@EXAMPLE.com
```

6.7. 映射解析器配置

Token 处理系统默认提供一个映射解析器。解析器名为 FilterMappingResolver。本节将涵盖其配

置。

注意

有关 映射 解析程序的一般信息，请参阅 红帽认证系统规划、安装和部署指南中的 映射解决问题部分。

6.7.1. key Set Mapping Resolver

在外部注册过程中，必须使用解析器解析密钥集，然后才能验证密钥。

键集映射解析器名称定义如下：

```
externalReg.mappingResolver=<keySet mapping resolver name>
```

例如：

```
externalReg.mappingResolver=keySetMappingResolver
```

以下配置示例显示了完整的实例配置：

```
mappingResolver.keySetMappingResolver.class_id=filterMappingResolverImpl
mappingResolver.keySetMappingResolver.mapping.0.filter.appletMajorVersion=0
mappingResolver.keySetMappingResolver.mapping.0.filter.appletMinorVersion=0
mappingResolver.keySetMappingResolver.mapping.0.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenATR=
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenCUID.end=a100000000000000000
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenCUID.start=a000000000000000000
mappingResolver.keySetMappingResolver.mapping.0.target.keySet=defKeySet
mappingResolver.keySetMappingResolver.mapping.1.filter.appletMajorVersion=1
mappingResolver.keySetMappingResolver.mapping.1.filter.appletMinorVersion=1
mappingResolver.keySetMappingResolver.mapping.1.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenATR=1234
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenCUID.end=
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenCUID.start=
mappingResolver.keySetMappingResolver.mapping.1.target.keySet=defKeySet
mappingResolver.keySetMappingResolver.mapping.2.filter.appletMajorVersion=
mappingResolver.keySetMappingResolver.mapping.2.filter.appletMinorVersion=
mappingResolver.keySetMappingResolver.mapping.2.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenATR=
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenCUID.end=
```

```
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenCUID.start=
mappingResolver.keySetMappingResolver.mapping.2.target.keySet=jForte
mappingResolver.keySetMappingResolver.mapping.order=0,1,2
```

上面的示例定义了三个映射，名为 0、1 和 2。它们使用 mappingResolver.keySetMappingResolver.mapping.order=0,1,2 行排序。这个顺序表示将首先针对映射过滤器 0 运行输入参数；只有在它们不匹配该过滤器时，才会在映射顺序中尝试下一个参数。例如，如果评估具有以下特征的令牌：

```
CUID=a000000000000000011
appletMajorVersion=0
appletMinorVersion=0
```

然后，它将传递映射 0 并为其分配目标（配置为 defKeySet ），因为 小应用程序版本匹配，并且 CUID 不在该映射的 CUID 启动和结束范围内。

另外，如果令牌具有以下参数：

```
CUID=b000000000000000000
ATR=2222
appletMajorVersion=1
appletMinorVersion=1
```

在这种情况下，此令牌无法映射 0， 因为它位于指定的 CUID 范围之外。它也会失败映射 1，因为应用程序版本匹配时 ATR 也会失败。以上令牌将分配给映射 2 及其目标 J Forte。

请注意，映射 2 如何为任何过滤器没有分配。这会导致映射与所有令牌匹配，并有效地使其成为"默认"值。像这样的映射顺序必须以映射顺序指定，因为不会评估任何其他映射。

6.7.2. 令牌类型(TPS)Mapping Resolver

Token Processing System 中定义的三个默认 tokenType 映射解析器：formatProfileMappingResolver、enrollProfileMappingResolver 和 pinResetProfileMappingResolver。与上一节中讨论的外部注册案例相比，内部注册案例实际是从定义的映射解析器计算的。

令牌类型映射解析器名称定义如下：

```
op.<op>.mappingResolver=<mapping resolver name>
```

例如：

```
op.enroll.mappingResolver=enrollProfileMappingResolver
```

以下配置示例描述了 enrollProfileMappingResolver ：

```
mappingResolver.enrollProfileMappingResolver.class_id=filterMappingResolverImpl
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.appletMajorVersion=1
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenCUID.end=b1000000000
000000000
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenCUID.start=b000000000
0000000000
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.0.target.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.appletMajorVersion=1
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenCUID.end=a0000000000
000001000
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenCUID.start=a000000000
0000000000
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenType=soKey
mappingResolver.enrollProfileMappingResolver.mapping.1.target.tokenType=soKey
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.appletMajorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenCUID.end=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenCUID.start=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenType=
mappingResolver.enrollProfileMappingResolver.mapping.2.target.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.order=1,0,2
```

在上例中为 register ProfileMappingResolver 定义了三个映射。映射名为 0、1 和 2。mappingResolver.enrollProfileMappingResolver.mapping.order=1,0,2 行定义处理映射的顺序。如果令牌与映射匹配，则不会对顺序进行进一步映射；如果与映射不匹配，就会尝试下一个映射。

如果令牌具有以下参数：

```
CUID=a000000000000000011
appletMajorVersion=1
appletMinorVersion=0
extension: tokenType=soKey
```

使用这个配置的令牌将与映射 1 的过滤器匹配，小应用程序版本匹配，CUID 在指定的开始和结束范

围内会失败，扩展名 tokenType 匹配。因此，此令牌将被分配该映射的目标 - soKey.

另外，如果令牌具有以下参数：

```
CUID=b00000000000000000010
appletMajorVersion=1
appletMinorVersion=1
```

在这种情况下，令牌将失败，因为 CUID 位于指定范围之外。然后它还将失败映射 0，因为缺少 tokenType 扩展。然后，此令牌将匹配映射 2，因为它没有指定的过滤器以匹配上一个过滤器的任何令牌。

6.8. 身份验证配置

Token 处理系统默认支持使用用户 ID 和密码(UidPwdDirAuthentication)进行基于目录的身份验证。身份验证实例使用以下模式在 CS.cfg 文件中定义：

```
auths.instance.<auths ID>.*
```

& lt;auths ID > 是验证首选项的 TPS 配置集引用的 authenticator 名称。例如：

```
op.enroll.userKey.auth.id=ldap1
```

以下配置示例显示了身份验证实例的完整定义：

```
auths.impl.UidPwdDirAuth.class=com.netscape.cms.authentication.UidPwdDirAuthentication
auths.instance.ldap1.pluginName=UidPwdDirAuth
auths.instance.ldap1.authCredName=uid
auths.instance.ldap1.dnpattern=
auths.instance.ldap1.externalReg.certs.recoverAttributeName=certsToAdd
auths.instance.ldap1.externalReg.cuidAttributeName=tokenCUID
auths.instance.ldap1.externalReg.tokenTypeAttributeName=tokenType
auths.instance.ldap1.ldap.basedn=dc=sjc,dc=example,dc=com
auths.instance.ldap1.ldap.ldapauth.authtype=BasicAuth
auths.instance.ldap1.ldap.ldapauth.bindDN=
auths.instance.ldap1.ldap.ldapauth.bindPWPrompt=ldap1
auths.instance.ldap1.ldap.ldapauth.clientCertNickname=subsystemCert cert-pki-tomcat
auths.instance.ldap1.ldap.ldapconn.host=host1.EXAMPLE.com
auths.instance.ldap1.ldap.ldapconn.port=389
auths.instance.ldap1.ldap.ldapconn.secureConn=False
auths.instance.ldap1.ldap.ldapconn.version=3
auths.instance.ldap1.ldap.maxConns=15
auths.instance.ldap1.ldap.minConns=3
```

```
auths.instance.ldap1.ldapByteAttributes=
auths.instance.ldap1.ldapStringAttributes=mail,cn,uid,edipi,pcc,firstname,lastname,exec-edipi,exec-pcc,exec-mail,certsToAdd,tokenCUID,tokenType
auths.instance.ldap1.ldapStringAttributes._000=###################################
auths.instance.ldap1.ldapStringAttributes._001=# For isExternalReg
auths.instance.ldap1.ldapStringAttributes._002=#   attributes will be available as
auths.instance.ldap1.ldapStringAttributes._003=#       $<attribute>$
auths.instance.ldap1.ldapStringAttributes._004=#   attributes example:
auths.instance.ldap1.ldapStringAttributes._005=#mail,cn,uid,edipi,pcc,firstname,lastname,exec-edipi,exec-pcc,exec-mail,certsToAdd,tokenCUID,tokenType
auths.instance.ldap1.ldapStringAttributes._006=###################################
auths.instance.ldap1.pluginName=UidPwdDirAuth
auths.instance.ldap1.ui.description.en=This authenticates user against the LDAP directory.
auths.instance.ldap1.ui.id.PASSWORD.credMap.authCred=pwd
auths.instance.ldap1.ui.id.PASSWORD.credMap.msgCred.extlogin=PASSWORD
auths.instance.ldap1.ui.id.PASSWORD.credMap.msgCred.login=password
auths.instance.ldap1.ui.id.PASSWORD.description.en=LDAP Password
auths.instance.ldap1.ui.id.PASSWORD.name.en=LDAP Password
auths.instance.ldap1.ui.id.UID.credMap.authCred=uid
auths.instance.ldap1.ui.id.UID.credMap.msgCred.extlogin=UID
auths.instance.ldap1.ui.id.UID.credMap.msgCred.login=screen_name
auths.instance.ldap1.ui.id.UID.description.en=LDAP User ID
auths.instance.ldap1.ui.id.UID.name.en=LDAP User ID
auths.instance.ldap1.ui.retries=3
auths.instance.ldap1.ui.title.en=LDAP Authentication
```

TPS 验证实例配置方式与 CA 的 UidPwdDirAuthentication 身份验证实例类似，因为它们都使用相同的插件处理。但是，TPS 在 CA 配置之上需要几个额外的参数。

如果是常见的操作（用于内部和外部注册），调用此验证方法的配置集允许 TPS 项目在客户端上标记 UID 和密码。这由上例中的 auths.instance.ldap1.ui.id.UID.name.en=LDAP User ID 和 auths.instance.ldap1.ui.id.PASSWORD.name.en=LDAP 密码 参数控制；此配置可向客户端显示 UID/password 对为"LDAP User ID"和"LDAP Password"。这两个参数都可以自定义。

credMap.authCred 条目配置内部身份验证插件如何接受与之显示的信息，以及 credMap.msgCred 条目配置如何将此信息传递给 TPS。这些字段允许您使用自定义插件实施，并且应保留其默认值，除非您使用自定义身份验证插件。

第 6.6 节 “外部注册” 中讨论与外部注册相关的参数。

与 CA 身份验证配置类似，您可以为相同的身份验证实施定义多个身份验证实例。当 TPS 服务于多个用户组时，这很有用；您可以指示每个组使用其自己的 TPS 配置集，每个配置都配置为使用其自己的目录服务器身份验证。

6.9. 连接器

连接器定义 TPS 如何与其他子系统（即 CA、KRA 和 TKS）通信。通常，这些参数会在 TPS 安装过程中设置。以下是连接器配置示例：

```
tps.connector.ca1.enable=true
tps.connector.ca1.host=host1.EXAMPLE.com
tps.connector.ca1.maxHttpConns=15
tps.connector.ca1.minHttpConns=1
tps.connector.ca1.nickName=subsystemCert cert-pki-tomcat
tps.connector.ca1.port=8443
tps.connector.ca1.timeout=30
tps.connector.ca1.uri.enrollment=/ca/ee/ca/profileSubmitSSLClient
tps.connector.ca1.uri.getcert=/ca/ee/ca/displayBySerial
tps.connector.ca1.uri.renewal=/ca/ee/ca/profileSubmitSSLClient
tps.connector.ca1.uri.revoke=/ca/ee/subsystem/ca/doRevoke
tps.connector.ca1.uri.unrevoke=/ca/ee/subsystem/ca/doUnrevoke
tps.connector.kra1.enable=true
tps.connector.kra1.host=host1.EXAMPLE.com
tps.connector.kra1.maxHttpConns=15
tps.connector.kra1.minHttpConns=1
tps.connector.kra1.nickName=subsystemCert cert-pki-tomcat
tps.connector.kra1.port=8443
tps.connector.kra1.timeout=30
tps.connector.kra1.uri.GenerateKeyPair=/kra/agent/kra/GenerateKeyPair
tps.connector.kra1.uri.TokenKeyRecovery=/kra/agent/kra/TokenKeyRecovery
tps.connector.tks1.enable=true
tps.connector.tks1.generateHostChallenge=true
tps.connector.tks1.host=host1.EXAMPLE.com
tps.connector.tks1.keySet=defKeySet
tps.connector.tks1.maxHttpConns=15
tps.connector.tks1.minHttpConns=1
tps.connector.tks1.nickName=subsystemCert cert-pki-tomcat
tps.connector.tks1.port=8443
tps.connector.tks1.serverKeygen=true
tps.connector.tks1.timeout=30
tps.connector.tks1.tksSharedSymKeyName=sharedSecret
tps.connector.tks1.uri.computeRandomData=/tks/agent/tks/computeRandomData
tps.connector.tks1.uri.computeSessionKey=/tks/agent/tks/computeSessionKey
tps.connector.tks1.uri.createKeySetData=/tks/agent/tks/createKeySetData
tps.connector.tks1.uri.encryptData=/tks/agent/tks/encryptData
```

TPS 配置集根据其 ID 指代这些连接器。例如：

```
op.enroll.userKey.keyGen.signing.ca.conn=ca1
```

可以定义相同类型的多个连接器（例如，多个 CA 连接器）。当一个 TPS 实例为不同令牌组提供多个后端证书系统服务器时，这很有用。

注意

目前不支持 TPS 中连接器的自动故障切换。需要执行手动故障切换过程，以将 TPS 指向备用 CA、KRA 或 TKS，只要它们是原始系统的克隆。

6.10. 撤销路由配置

要配置撤销路由，您必须首先定义相关 CA 连接器列表，并以以下格式将它们添加到连接器列表中：

```
tps.connCAList=ca1,ca2
```

另外，您必须在 TPS nssdb 中添加 CA 签名证书并设置信任：

```
#cd <TPS instance directory>/alias
```

```
#certutil -d . -A -n <CA signing cert nickname> -t “CT,C,C” -i <CA signing cert b64 file name>
```

最后，必须使用以下选项将 CA 签名证书的别名添加到连接器中：

```
tps.connector.ca1.caNickname=caSigningCert cert-pki-tomcat CA
```

注意

在 CA 发现过程中，TPS 可以自动计算 CA 的授权密钥标识符并将其添加到连接器配置中。例如：

```
tps.connector.ca1.caSKI=i9wOnN0QZLkzkndAB1MKMcjbRP8=
```

这个行为是预期的。

6.11. 设置服务器端密钥生成

服务器端密钥生成意味着密钥由密钥恢复授权(KRA)生成，它是一个可选的证书系统子系统。需要 KRA 生成密钥，以允许在丢失或已损坏的令牌时恢复密钥，或者在外部注册时检索密钥。这部分论述了如何在 TMS 中配置服务器端密钥生成。

在 TPS 安装过程中，要求您指定您是否要使用密钥归档。如果您确认，设置将执行自动基本配置，特

别是以下参数：

KRA 的 TPS 连接器参数：

```
tps.connector.kra1.enable=true
tps.connector.kra1.host=host1.EXAMPLE.com
tps.connector.kra1.maxHttpConns=15
tps.connector.kra1.minHttpConns=1
tps.connector.kra1.nickName=subsystemCert cert-pki-tomcat
tps.connector.kra1.port=8443
tps.connector.kra1.timeout=30
tps.connector.kra1.uri.GenerateKeyPair=/kra/agent/kra/GenerateKeyPair
tps.connector.kra1.uri.TokenKeyRecovery=/kra/agent/kra/TokenKeyRecovery
```

用于服务器端密钥生成的 TPS 配置集参数：

```
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true
```

设置 serverKeygen.enable=true 选项，使 serverKeygen.archive 生效。

重要

LunaSA HSM 不支持 RSA 加密时比 2048 字节小的密钥大小。

例如：要将密钥大小为 2048 字节，请在 /var/lib/pki/instance_name/tps/conf/CS.cfg 文件中设置以下参数：

```
op.enroll.userKey.keyGen.encryption.keySize=2048
```

TKS 配置：

以下配置了用于 TKS 和 KRA（通过 TPS）通信的传输证书的别名：

```
tks.drm_transport_cert_nickname=transportCert cert-pki-tomcat KRA
```

引用的传输证书还必须存在于 TKS 实例安全模块中。例如：

```
transportCert cert-pki-tomcat KRA                    u,u,u
```

KRA 配置

根据 PKCS#11 令牌，参数 kra.keygen.temporaryPairs、kra.keygen.sensitivePairs 和 kra.keygen.keygen.extractablePairs 可以为密钥生成选项自定义。这些参数默认设置为 false。

这些参数的以下值已使用 Red Hat Certificate System 支持的一些安全模块进行了测试：

NSS（在 FIPS 模式中）：

```
kra.keygen.extractablePairs=true
```

nCipher nShield Connect 6000（默认情况下，不指定工作）：

指定 RSA 密钥：

```
kra.keygen.temporaryPairs=true
```

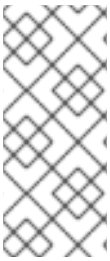
（不要指定任何其他参数。）

生成 ECC 密钥：

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=false
kra.keygen.extractablePairs=true
```

LunaSA CKE - 密钥导出模型（非FIPS 模式）：

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=true
kra.keygen.extractablePairs=true
```



注意

Gemalto SafeNet LunaSA 只支持 CKE - 密钥导出模型中的 PKI 私钥提取，且仅在非 FIPS 模式中支持。LunaSA Cloning 模型和 FIPS 模式中的 CKE 模型不支持 PKI 私钥提取。

注意

当 LunaSA CKE - Key Export Model 处于 FIPS 模式时，无法提取 pki 私钥。

6.12. 设置新密钥设置

这部分论述了在 Token Processing System(TPS)和 Token Key Service(TKS)中设置的默认密钥的替代选择。

TKS 配置

在 TKS 中使用 /var/lib/pki/instance_name/tks/conf/CS.cfg 文件中的以下选项配置默认密钥：

```
tks.defKeySet._000=##
tks.defKeySet._001=## Axalto default key set:
tks.defKeySet._002=##
tks.defKeySet._003=## tks.defKeySet.mk_mappings.#02#01=<tokenname>:<nickname>
tks.defKeySet._004=##
tks.defKeySet.auth_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.kek_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.mac_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.nistSP800-108KdfOnKeyVersion=00
tks.defKeySet.nistSP800-108KdfUseCuidAsKdd=false
```

以上配置定义了特定于在 TMS 中可以使用的某些类型或类的设置。最重要的部分是 3 开发人员或（开箱即用）会话密钥，它们用于在对称密钥移交前创建安全频道。其他类型的键对这些键可能有不同的默认值。

描述 nistSP800 键 实用程序方法的设置控制是否使用了此方法。具体来说，tks.defKeySet.nistSP800-108KdfOnKeyVersion 选项的值决定了将使用 NIST 版本。nistSP800-108KdfUseCuidAsKdd 选项允许您在处理过程中使用传统的密钥 ID 值 CUID。较新的 KDD 值最常被使用，因此默认禁用这个选项(false)。这可让您配置新密钥集，以启用对新类密钥的支持。

例 6.2. 为 jForte 类启用支持

要启用对 jForte 类的支持，请设置：

```
tks.jForte._000=##
tks.jForte._001=## SAFLink's jForte default key set:
tks.jForte._002=##
tks.jForte._003=## tks.jForte.mk_mappings.#02#01=<tokenname>:<nickname>
tks.jForte._004=##
tks.jForte.auth_key=#30#31#32#33#34#35#36#37#38#39#3a#3b#3c#3d#3e#3f
tks.jForte.kek_key=#50#51#52#53#54#55#56#57#58#59#5a#5b#5c#5d#5e#5f
tks.jForte.mac_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.jForte.nistSP800-108KdfOnKeyVersion=00
tks.jForte.nistSP800-108KdfUseCuidAsKdd=false
```

> 请注意，与上例相比，3 静态会话键的区别。

证书系统支持 Secure Channel Protocol 03(SCP03)用于 Giesecke & Devrient(G&D)Smart Cafe 6 智能卡。要在 TKS 中启用对那些智能卡的 SCP03 支持，请在 /var/lib/pki/instance_name/tks/conf/CS.cfg 文件中设置：

```
tks.defKeySet.prot3.divers=emv
tks.defKeySet.prot3.diversVer1Keys=emv
tks.defKeySet.prot3.devKeyType=DES3
tks.defKeySet.prot3.masterKeyType=DES3
```

TPS 配置

当支持的客户端试图对令牌执行操作时，必须将 TPS 配置为识别新密钥集。默认 defKeySet 最常被使用。

决定 TPS 中的 keySet 的主要方法涉及 第 6.7 节 “映射解析器配置”。有关建立此解析器机制所需的具体设置，请查看链接部分。

如果 KeySet Mapping Resolver 不存在，则 TPS 提供了几个回退方法来确定正确的 keySet ：

- 您可以将 tps.connector.tks1.keySet=defKeySet =defKeySet 添加到 TPS 的 CS.cfg 配置文件中。
- 某些客户端可以配置为显式传递所需的 keySet 值。但是，企业安全客户端在此刻没有此功能。
- 当 TPS 根据所需方法计算正确的 keySet 时，到 TKS 的所有请求来帮助创建安全频道传递 keySet 值。然后 TKS 可以使用自己的 keySet 配置（上述步骤）来确定如何继续。

6.13. 设置新主密钥

本节将描述在 Token Key Service(TKS)中设置新主密钥所需的步骤和配置。有关背景信息，请参阅 红帽认证系统规划、安装和部署指南。

过程 6.1. 创建新主密钥

1. 获取访问 **TKS** 安全数据库所需的内部 **PIN**：

```
# cat /var/lib/pki/pki-tomcat/tks/conf/password.conf
internal=649713464822
internaldb=secret12
replicationdb=-752230707
```

2. 打开 **TKS** 实例的别名/目录：

```
# cd /var/lib/pki/pki-tomcat/alias
```

3. 使用 **tkstool** 程序生成新的 **master** 密钥。例如：

```
# tkstool -M -n new_master -d /var/lib/pki/pki-tomcat/alias -h <token_name>
Enter Password or Pin for "NSS Certificate DB":

Generating and storing the master key on the specified token . . .

Naming the master key "new_master" . . .

Computing and displaying KCV of the master key on the specified token . . .

new_master key KCV:  CA5E 1764
```

4. 验证密钥是否已正确添加到数据库中：

```
# tkstool -L -d .


 slot:  NSS User Private Key and Certificate Services
token:  NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
      <0> new_master
```

6.13.1. 生成和传输主密钥（主要 Ceremony）

如果主密钥将用于外部令牌或多个位置，则必须进行 嵌套，以便安全地传输到硬件令牌。tkstool 实用程序可用于生成传输密钥，然后用于将主密钥发送到生成令牌的设备。传输嵌套的 master 密钥的进程通常称为密钥 Ceremony。

注意

传输密钥只能用于生成的主密钥。

过程 6.2. 生成并传输 Wrapped 主密钥

1. 获取访问 **Token Key Service** 安全数据库所需的内部 **PIN**：

```
# cat /var/lib/pki/pki-tomcat/tks/conf/password.conf

internal=649713464822
internaldb=secret12
replicationdb=-752230707
```

2. 打开 **TKS** 实例 别名/ 目录：

```
# cd /var/lib/pki/pki-tomcat/alias
```

3. 创建名为 **transport** 的 传输 密钥：

```
# tkstool -T -d . -n transport
```

注意

tkstool 程序输出每个生成的三个会话键的密钥共享和 **KCV** 值。将这些文件保存到文件中，因为稍后会在这个流程中的新数据库中重新生成传输密钥，并在丢失密钥时重新生成密钥。

4. 出现提示时，填写数据库密码。然后，根据屏幕的说明生成随机 **seed**。

```
A random seed must be generated that will be used in the
creation of your key.  One of the easiest ways to create a
random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter
is full.  DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!


Continue typing until the progress meter is full:

|************************************************************|
```

```
Finished.


Type the word "proceed" and press enter
```

5. 下一提示将生成一系列会话密钥。按照屏幕说明操作，直到最终消息：

```
Successfully generated, stored, and named the transport key!
```

6. 使用 transport 密钥生成并打包一个 master 密钥，并将其存储在名为 的文件 ：

```
# tkstool -W -d . -n new_master -t transport -o file
Enter Password or Pin for "NSS Certificate DB":
Retrieving the transport key (for wrapping) from the specified token . . .
Generating and storing the master key on the specified token . . .
Naming the master key "new_master" . . .
Successfully generated, stored, and named the master key!
Using the transport key to wrap and store the master key . . .
Writing the wrapped data (and resident master key KCV) into the
 file called "file" . . .

    wrapped data:   47C0 06DB 7D3F D9ED
                    FE91 7E6F A7E5 91B9
    master key KCV: CED9 4A7B
    (computed KCV of the master key residing inside the wrapped data)
```

7. 将嵌套的 master 密钥复制到适当的位置或工具中。

8. 如有必要，在 HSM 或设备中生成新安全数据库：

```
# tkstool -N -d <directory>
```

或者，添加 -I 选项来生成与最初在新数据库中生成的键相同。以这种方式重新生成 transport 密钥需要您输入会话密钥共享和 KCV，用于这个过程早期生成的每个会话密钥。

```
# tkstool -I -d <directory> -n verify_transport
```

9. 使用 transport 密钥来取消包装存储在 文件中的主密钥。提示时提供安全数据库 PIN：

```
# tkstool -U -d directory -n new_master -t verify_transport -i file
Enter Password or Pin for "NSS Certificate DB":
```

```
Retrieving the transport key from the specified token (for
 unwrapping) . . .
Reading in the wrapped data (and resident master key KCV) from
 the file called "file" . . .

    wrapped data:   47C0 06DB 7D3F D9ED
                    FE91 7E6F A7E5 91B9
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to temporarily unwrap the master key to
recompute its KCV value to check against its pre-computed KCV value . . .
    master key KCV: CED9 4A7B
    (computed KCV of the master key residing inside the wrapped data)
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to unwrap and store the master key on the
 specified token . . .
Naming the master key "new_master" . . .
Successfully unwrapped, stored, and named the master key!
```

10. 验证密钥是否已正确添加到数据库中：

```
# tkstool -L -d
slot:  NSS User Private Key and Certificate Services
token:  NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
   <0> transport
   <1> new_master
```

6.14. 设置 TKS/TPS SHARED SYMMETRIC KEY

共享对称密钥必须存在于 TPS 和 TKS 子系统的 NSS 数据库中。创建 TPS 子系统时会自动生成此密钥。如果同一 Tomcat 实例中都安装了 TPS 和 TKS，则不需要额外的设置，因为 TKS 会自动使用 TPS 创建的密钥；但是，如果两个子系统位于单独的实例上，或者根据这部分描述的步骤，必须按照本小节中介绍的步骤安全地将密钥传输到 TKS。

有几种可能的方法可用于安全地传输 TPS 和 TKS 之间的共享密钥：

- authomatic 方法：当 TPS 的子系统证书保存在软件 NSS 数据库中时，此方法可以正常工作。

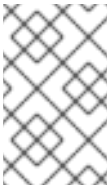
- 如果上述方法失败，可以使用一个回退手动方法，其中使用 tkstool 程序在 TPS 上生成共享密钥，这样可将密钥从 TPS 嵌套，在不在传输中公开密钥的情况下进行安全传输，并取消将其嵌

套在 TKS NSS 数据库中。

下面描述了 TPS 和 TKS 的一般配置，无论用于导入密钥的方法如何。请注意，自动方法会自动生成这些配置。

TKS

```
tks.useNewSharedSecretNames=true
tps.0.host=dhcp-16-206.sjc.example.com
tps.0.nickname=TPS-<tps host name>-8443 sharedSecret
tps.0.port=8443
tps.0.userid=,TPS-<tps host name>-8443
tps.list=0
```

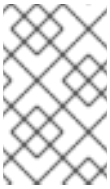


注意

当一个 TKS 连接到多个 TPS 实例时，上述列表可以扩展。

TPS

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```



注意

主机名必须与 TKS 端配置的名称相同。

6.14.1. 手动生成和传输共享统计密钥

这部分论述了如何手动生成和传输共享对称密钥。如果自动生成和传输失败时，这种方法很有用，但在其他情况下应避免。

manual 方法包含两个流程。第一个是在 Token Key Service 一侧执行，第二个在令牌处理系统中执行。

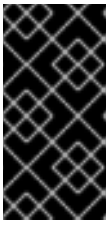
过程 6.3. Manual Shared Secret Key Method - TKS side

1. 在第一个系统上安装令牌密钥服务。有关安装说明，请参阅 *红帽认证系统规划、安装和部署指南*。

2. ***停止 TKS 服务：***

```
#systemctl stop pki-tomcatd@pki-tomcat.service
```

3. ***更改到 /var/lib/pki/pki-tomcat/alias 目录，并使用 tkstool 在 TKS 上创建共享密钥。在重启新的 TKS 实例前，请确保生成共享密钥。***



重要

tkstool 脚本将在密钥创建过程中显示密钥的相关信息。确保记下这些信息，因为稍后需要将密钥导入到 TPS。

```
#cd /var/lib/pki/pki-tomcat/alias
#tkstool -T -d /var/lib/pki/pki-tomcat/tks/alias -n TPS-<tps host name>-8443 sharedSecret
Generating the first session key share . . .
   first session key share:      792F AB89 8989 D902
                                 9429 6137 8632 7CC4
   first session key share KCV:  D1B6 14FD
Generating the second session key share . . .
   second session key share:     4CDF C8E0 B385 68EC
                                 380B 6D5E 1C19 3E5D
   second session key share KCV:  1EC7 8D4B
Generating the third session key share . . .
   third session key share:      CD32 3140 25B3 C789
                                 B54F 2C94 26C4 9752
   third session key share KCV:  73D6 8633
Generating first symmetric key . . .
Generating second symmetric key . . .
Generating third symmetric key . . .
Extracting transport key from operational token . . .
   transport key KCV:  A8D0 97A2
Storing transport key on final specified token . . .
Naming transport key "sharedSecret" . . .
Successfully generated, stored, and named the transport key!
```

4. ***在 TKS 中配置新密钥：***

```
tks.useNewSharedSecretNames=true
tps.0.host=dhcp-16-206.sjc.redhat.com
tps.0.nickname=TPS-<tps host name>-8443 sharedSecret
tps.0.port=8443
tps.0.userid=TPS-<tps host name>-8443 sharedSecret
tps.list=0
```

5. 启动 **TKS**：

```
#systemctl start pki-tomcatd@pki-tomcat.service
```

过程 6.4. 手动共享 secret 密钥方法 - TPS 侧

1. 在第二个系统上安装令牌处理系统。有关安装说明，请参阅 *红帽认证系统 9 规划、安装和部署指南*。

2. 停止 **TPS** 服务：

```
#systemctl stop pki-tomcatd@pki-tomcat.service
```

3. 更改到 **/var/lib/pki/pki-tomcat/alias** 目录，并使用 **tkstool** 将共享密钥导入到 **NSS** 软件令牌中：

```
#cd /var/lib/pki/pki-tomcat/alias
#tkstool -I -d . -n TPS-<tps host name>-8443 sharedSecret
```

此时，脚本会提示您生成并嵌套在上述流程中的 **TKS** 一侧显示的会话密钥共享。

4. 在 **TPS** 中配置共享 **secret**：

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```

5. 启动 **TPS** 服务：

```
#systemctl start pki-tomcatd@pki-tomcat.service
```

6.15. 将不同的 APPLETS 用于不同的 SCP 版本

在证书系统中，**/var/lib/instance_name/tps/conf/CS.cfg** 文件中的以下参数指定应该为每个令牌操作载入哪个小应用程序：

```
op.operation.token_type.update.applet.requiredVersion=version
```

但是，您还可以通过添加以下参数来为特定的 SCP 版本设置单独的应用程序：

```
op.operation.token_type.update.applet.requiredVersion.prot.protocol_version=version
```

证书系统支持为以下操作设置单独的协议版本：

- 格式
- enroll
- pinReset

例 6.3. 为注册操作设置协议版本

在为 userKey 令牌执行注册操作时，要为 SCP03 和所有其他协议配置特定的小应用程序：

1. 编辑 /var/lib/instance_name/tps/conf/CS.cfg 文件：

   a. 设置 op.enroll.userKey.update.applet.requiredVersion 参数，以指定默认使用的 小程序。例如：

   ```
   op.enroll.userKey.update.applet.requiredVersion=1.4.58768072
   ```

   b. 设置 op.enroll.userKey.update.applet.requiredVersion.prot.3 参数，以配置应用程序证书系统用于 SCP03 协议。例如：

   ```
   op.enroll.userKey.update.applet.requiredVersion.prot.3=1.5.558cdcff
   ```

2. 重启证书系统：

   ```
   systemctl restart pki-tomcatd@instance_name.service
   ```

有关为 Giesecke 和 Devrient(G&D)智能 Cafe 6 智能卡在 TKS 中启用 SCP03 的详情，请参考第 6.12 节 “设置新密钥设置”。

第 7 章 撤销证书和发布 CRL

CertificateCertificate Systemnbsp;System 提供了撤销证书的方法，以及生成撤销的证书列表，称为证书撤销列表(CRL)。本章论述了撤销证书的方法，描述了 CMC 撤销调用，并提供有关 CRL 和设置 CRL 的详情。

7.1. 关于撤销证书

证书可以被最终用户（证书的原始所有者）或证书管理器代理撤销。最终用户可以使用在端点页面中提供的撤销表单来撤销证书。代理可以使用代理服务接口中的适当形式来撤销最终用户证书。这两种情况下都需要使用基于证书的（SSL/TLS 客户端身份验证）。

最终用户只能撤销包含与用于身份验证的证书相同的主题名称的证书。身份验证成功后，服务器会列出属于最终用户的证书。然后，最终用户可以选择被撤销的证书，也可以撤销列表中的所有证书。最终用户还可指定其他详情，如每个证书的撤销日期和撤销原因，或针对整个列表指定列表。

代理可以基于一系列序列号或基于主题名称组件来撤销证书。提交撤销请求时，代理会收到一个证书列表，从中可以选择一个要撤销的证书。有关代理如何撤销终止证书的说明，请参阅 *Red Hat 证书系统 9 规划、安装和部署指南*。

撤销请求后，证书管理器会将其内部数据库中对应的证书记录标记为已撤销，如果配置为这样做，则从发布目录中删除已撤销的证书。这些更改反映在 CA 问题的下一个 CRL 中。

使用公钥证书的服务器和客户端应用程序作为 ID 令牌需要访问证书的有效性。由于确定证书的有效性的因素之一就是其撤销状态，所以这些应用程序需要了解要被验证的证书是否已被撤销。CA 负责执行以下操作：

- 如果 CA 收到了撤销请求并批准，则撤销证书。
- 将撤销的证书状态提供给需要验证其有效状态的第三方或应用程序。

每当证书被撤销时，证书管理器都会自动更新其内部数据库中证书的状态，它会将其内部数据库中的证书副本标记为已撤销状态并从发布目录中移除证书。如果证书管理器被配置为从数据库中删除证书。

用于分发证书的标准方法之一是发布已撤销的证书列表，称为证书撤销列表(CRL)。CRL 是一个公开可用证书列表，已撤销。

证书管理器可以被配置为生成 CRL。通过在 CRL 配置中启用特定于扩展的模块，可以创建这些 CRL 以符合 X.509 标准。服务器通过其 CRL 发出点框架支持标准 CRL 扩展；有关为发出点设置 CRL 扩展的更多信息，请参阅 第 7.3.3 节 “设置 CRL 扩展”。证书管理器可以在每次证书被撤销和定期间隔时生成 CRL。如果设置了发布，则 CRL 可以发布到文件、LDAP 目录或 OCSP 响应程序。

发布 CRL 并由发布 CRL 中列出的证书的 CA 发布或由该 CA 授权的实体发布并签名以发布 CRL 的实体。CA 可使用单个密钥对为证书和 CRL 签名它，或两个单独的密钥对，一个用于签名 CRL，另一个用于签名 CRL。

默认情况下，证书管理器使用一个密钥对对它生成的证书和 CRL 进行签名。要为证书管理器创建另一个密钥对，并专门用于签名 CRL，请参阅 第 7.3.4 节 “将 CA 设置为使用其他证书来签名 CRL”。

在签发点时生成 CRL，并配置了启用了 CRL 生成的时间。

启用 CRL 后，服务器会在撤销证书时收集撤销信息。服务器会尝试将撤销的证书与设置的所有发布点匹配。给定证书无法匹配任何发布点、其中一个发布点、多个发布点或所有发布点。当被撤销的与发布点匹配的证书时，服务器会将有关证书的信息存储在发出点的缓存中。

缓存会复制到内部目录，并以为复制缓存设定的时间间隔。当达到创建 CRL 的时间间隔时，会从缓存创建一个 CRL。如果为此发布点设置了 delta CRL，此时也会创建一个 delta CRL。完整的 CRL 包含所有已撤销的证书信息，因为证书管理器开始收集这些信息。delta CRL 包含自完整 CRL 最后一次更新以来所有已撤销的证书信息。

完整的 CRL 按顺序编号，如 delta CRL。完整 CRL 和 delta CRL 可以有相同的数字；在这种情况下，delta CRL 的编号与 下一个 完整 CRL 相同。例如，如果完整的 CRL 是第一个 CRL，则代表 CRL 1。delta CRL 为 Delta CRL 2。CRL 1 和 Delta CRL 2 中的数据与下一个完整的 CRL 相同，即 CRL 2。

注意

当修改发布点的扩展时，不会使用发出该点的下一个完整 CRL 创建 delta CRL。使用创建 的第二个 完整 CRL 创建 delta CRL，然后所有后续完整 CRL。

内部数据库仅存储最新的 CRL 和 delta CRL。在创建每个新 CRL 时，旧的 CRL 将被覆盖。

当发布 CRL 时，对 CRL 和 delta CRL 每次更新都会发布到发布集合中指定的位置。发布方法决定了存储了多少个 CRL。对于文件发布，每个 CRL 使用 CRL 数量发布到文件中，因此不会覆盖任何文件。对于 LDAP 发布，发布的每个 CRL 都会替换包含目录条目中的 CRL 属性中的旧 CRL。

默认情况下，CRL 不包含关于已撤销过期证书的信息。服务器可通过为发出点启用该选项，包括撤销的过期证书。如果包含过期的证书，当证书过期时，不会从 CRL 中删除关于已撤销证书的信息。如果没有包含过期的证书，则在证书过期时从 CRL 中删除有关已撤销证书的信息。

7.1.1. 用户初始化的调用

当最终用户提交证书撤销请求时，撤销过程中的第一步就是证书管理器来识别和验证最终用户，以验证用户是否尝试撤销自己的证书，而不是属于他人的证书。

在 SSL/TSL 客户端身份验证中，服务器需要最终用户提供与要撤销的主题名称相同的证书，并将其用于身份验证目的。服务器通过在内部数据库中为客户端身份验证的证书中的主体名称映射，来验证撤销请求的真实性。只有在证书映射到其内部数据库中的一个或多个有效或过期证书时，服务器才会撤销证书。

身份验证成功后，服务器会列出与为客户端身份验证提供的证书的主题名称相匹配的有效或过期的证书。然后用户可以选择要撤销的证书或撤销列表中所有证书。

7.1.2. 撤销证书的原因

证书管理器可以撤销其发出的任何证书。通常，撤销 CRL 中通常包含的证书的接受原因代码，如下所示：

- 0.Unspecified; 没有特定原因。
- 1.与证书关联的私钥已被破坏。
- 2.与发布证书的 CA 关联的私钥已被破坏。
- 3.证书的所有者不再与证书颁发者相关，并且没有证书获得的访问权限或不再需要它的权限。
- 4.另一个证书取代了这个证书。

- 5.发布证书的 CA 已创建要操作的 CA。

- 6.该证书正在保存等待的进一步操作。它被视为已撤销，但将来可能会发生，以便证书再次激活并有效。

- 8.该证书将从 CRL 中删除，因为它已从拥有中移除。这只适用于 delta CRL。

- 9.该证书会被撤销，因为证书所有者的权限已被撤回。

证书可以被管理员、代理和最终实体撤销。带有代理权限的代理和管理员可使用代理服务页面中的表单撤销证书。最终用户可以使用端点接口的 Revocation 选项卡中的表单来撤销证书。最终用户只能撤销自己的证书，而代理和管理员可以撤销服务器发布的任何证书。为了撤销证书，还需要最终用户对服务器进行身份验证。

每当撤销证书时，证书管理器会在其内部数据库中更新证书的状态。服务器使用内部数据库中的条目来跟踪所有已撤销的证书，并在配置后，通过向中央存储库发布 CRL，以通知列表中的其他用户不再有效。

7.1.3. CRL 签发点

由于 CRL 可能会增长非常大，因此有几种方法可以尽量减少检索和交付大型 CRL 的开销。这些方法之一会对整个证书空间进行分区，并将一个单独的 CRL 与每个分区相关联。此分区称为 CRL 发布点，即维护所有已撤销证书的子集的位置。分区可以基于被撤销的证书是 CA 证书，无论是出于特定原因被撤销，还是使用特定配置集发布的。每个发布点都由其名称来标识。

默认情况下，证书管理器生成并发布单个 CRL，即 master CRL。发行点可为所有证书生成 CRL，仅适用于 CA 签名证书，或为所有证书（包括过期的证书）生成 CRL。

定义了发布点后，它们可以包含在证书中，以便需要检查证书的撤销状态的应用程序可以访问证书中指定的 CRL 发出点，而不是 master 或主 CRL。由于在发布点上维护的 CRL 比 master CRL 小，因此检查撤销状态更快。

通过设置 CRLDistributionPoint 扩展，可将 CRL 分发点与证书关联。

7.1.4. Delta CRLs

可以为任何定义的发布点发布 delta CRL。delta CRL 包含有关从最后更新到完整 CRL 后撤销的任何证书的信息。通过启用 DeltaCRLIndicator 扩展来创建发布点的 delta CRL。

7.1.5. 发布 CRL

证书管理器可将 CRL 发布到文件、LDAP 兼容目录或 OCSP 响应者。在证书管理器中发布 CRL 的位置和频率，如 第 8 章 发布证书和 CRL 所述。

因为 CRL 可能非常大，所以发布 CRL 可能需要很长时间，所以可能会中断进程。可以将特殊发布程序配置为通过 HTTP1.1 将 CRL 发布到文件，如果进程中断，则 CA 子系统的 Web 服务器可以在其中断时恢复发布，而不必再次开始发布。第 8.8 节 “设置恢复 CRL 下载” 中描述的。

7.1.6. 证书调用页

证书管理器的端点页面包括默认 HTML 表单，用于撤销通过 SSL/TLS 客户端验证的调用。表单可从 Revocation 标签页访问。您可以点击 User Certificate 链接来查看此类撤销的表单。

要更改表单外观来满足机构的要求，请编辑 UserRevocation.html，它的格式允许 SSL/TSL 客户端被验证地撤销客户端或个人证书。该文件位于 /var/lib/instance_name/webapps/subsystem_type/ee/subsystem_type 目录中。

7.2. 执行 CMC REVOCATION

与证书管理 over CMS(CMC)注册类似，CMC revocation 允许用户设置撤销客户端，并使用代理证书或具有匹配 subjectDN 属性的用户证书签署撤销请求。然后用户可以向证书管理器发送签名请求。

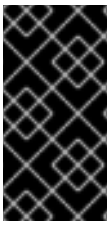
另外，CMC 撤销程序也可以使用 Shared Secret Token 机制进行身份验证。详情请参阅 红帽认证系统规划、安装和部署指南。

无论用户或代理是否为请求签名，或者是否使用了共享 Secret 令牌，证书管理器会在收到有效撤销请求时自动撤销证书。

证书系统为 CMC 撤销请求提供以下工具：

- CMCRequest.详情请查看 第 7.2.1 节 “使用 CMCRequest撤销证书”。

- CMCRevoke.详情请查看 第 7.2.2 节 “使用 CMCRevoke撤销证书”。



重要

红帽建议使用 CMCRequest 工具生成 CMC 重新调用请求，因为它提供了大于 CMCRevoke 的选项。

7.2.1. 使用 CMCRequest撤销证书

使用 CMCRequest 撤销证书：

1. 为 CMC 撤销请求创建一个配置文件，如 /home/user_name/cmc-request.cfg，其中包含以下内容：

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#output: full path for the CMC request in binary format
output=/home/user_name/cmc.revoke.userSigned.req

#tokenname: name of token where user signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for user signing certificate which will be used
#to sign the CMC full request.
nickname=signer_user_certificate

#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert8.db which stores the user signing
#certificate and keys
password=myPass

#format: request format, either pkcs10 or crmf.
format=pkcs10

## revocation parameters
revRequest.enable=true
revRequest.serial=45
revRequest.reason=unspecified
revRequest.comment=user test revocation
revRequest.issuer=issuer
revRequest.sharedSecret=shared_secret
```

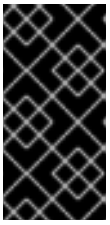
2. *创建 **CMC** 请求：*

```
# CMCRequest /home/user_name/cmc-request.cfg
```

*如果命令成功，则 **CMCRequest** 程序会在请求配置文件的 **output** 参数中指定的文件中存储 **CMC** 请求。*

3. *创建配置文件，如 **/home/user_name/cmc-submit.cfg**，您可以在稍后的步骤中使用该文件，将 **CMC** 撤销请求提交到 **CA**。在创建的文件中添加以下内容：*

```
#host: host name for the http server
host=>server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be
#in binary format
input=/home/user_name/cmc.revoke.userSigned.req

#output: full path for the response in binary format
output=/home/user_name/cmc.revoke.userSigned.resp

#tokenname: name of token where SSL client authentication certificate
#can be found (default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/

#clientmode: true for client authentication, false for no client
#authentication. This parameter will be ignored if secure=false
clientmode=true

#password: password for cert8.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=signer_user_certificate
```



重要

如果 CMC 撤销请求已签名，则将 secure 和 clientmode 参数设置为 true，再填充 nickname 参数。

4. 根据对请求签名的，Http Client 配置文件中的 servlet 参数必须相应地设置：

- 如果代理对请求进行了签名，请设置：

```
servlet=/ca/ee/ca/profileSubmitCMCFull
```

- 如果用户签名了请求，请设置：

```
servlet=/ca/ee/ca/profileSubmitSelfSignedCMCFull
```

5. 提交 CMC 请求：

```
# HttpClient /home/user_name/cmc-submit.cfg
```

有关使用 CMCRequest 撤销证书的详情，请查看 CMCRequest(1) man page。

7.2.2. 使用 CMCRevoke撤销证书

CMC 撤销程序 CMCRevoke 用于通过代理的证书为撤销请求签名。这个实用程序只传递必要的信息 - 证书序列号、签发者名称和撤销原因 - 来标识要撤销的证书，然后标识执行撤销的 CA 代理（certificate nickname 和带有证书的数据库）。

正在撤销证书的原因可以是以下任意一种（带有传递给 CMCRevoke 实用程序的值）：

- 0 - 未指定
- 1 - 密钥已被破坏

- *2 - CA 密钥已被破坏*
- *3 - 员工附属机构已改变*
- *4 - 证书已被替换*
- *5 - 操作考虑*
- *6 - 证书位于*

有关可用工具参数的详细信息，请参阅 命令行工具指南。

7.2.2.1. 测试 CMCRevoke

1. *为现有证书创建 CMC 撤销请求。*

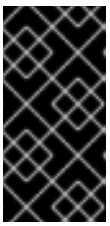
```
CMCRevoke -d/path/to/agent-cert-db -nnickname -iissuerName -sserialName -mreason -ccomment
```

例如，如果包含代理证书的目录是 ~jsmith/.mozilla/firefox/，则证书的别名为 AgentCert，并且命令的序列号为 22，如下所示：

```
CMCRevoke -d"~jsmith/.mozilla/firefox/" -n"ManagerAgentCert" -i"cn=agentAuthMgr" -s22 -m0 -c"test comment"
```

注意

在引号中包含空格的值会包括在引号中。



重要

在 参数及其值之间没有空格。例如，指定序列号为 26 的序列号为 -s26，而不是 -s 26。

2. 打开"终端"页面。

```
https://server.example.com:8443/ca/ee/ca
```

3. 选择 Revocation 选项卡。

4. 选择菜单上的 CMC Revoke 链接。

5. 将 CMCRevoke 的输出粘贴到文本区域。

6. 从粘贴的内容中删除 -----BEGIN NEW CERTIFICATE REQUEST----- 和 ----END NEW CERTIFICATE REQUEST-----。

7. 点 Submit。

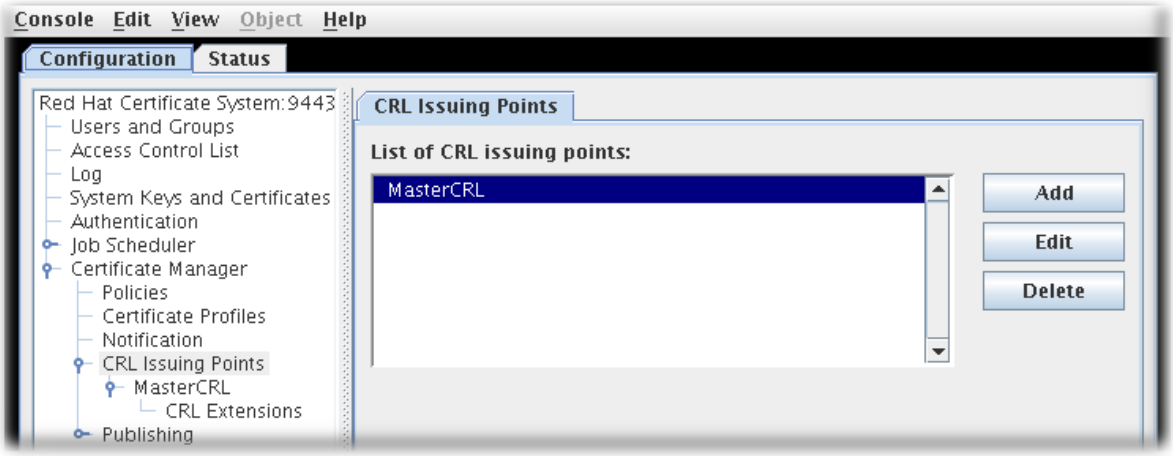
8. 返回后的页面应确认已撤销了正确的证书。

7.3. 发出 CRL

1. 证书管理器使用其 CA 签名证书密钥来签署 CRL。要为 CRL 使用单独的签名密钥对，请设置 CRL 签名密钥并更改证书管理器配置，以使用此密钥为 CRL 签名。如需更多信息，请参阅 第 7.3.4 节 “将 CA 设置为使用其他证书来签名 CRL”。

2. 设置 CRL 发行点。已经为 master CRL 设置并启用了一个发布点。

图 7.1. 默认 CRL 发行点



可以创建 CRL 的额外发布点。详情请查看 第 7.3.1 节 “配置发行得分”。

根据配置发布点时所设置的选项，还会有五种 CRL 的 CRL 类型来定义 CRL 将列出的内容：

- *Master CRL 包含来自整个 CA 的已撤销证书的列表。*
- *ARL 是一个授权撤销列表，仅包含已撤销的 CA 证书。*
- *带有过期证书的 CRL 包括撤销在 CRL 中过期的证书。*
- *证书配置集中的 CRL 决定基于最初创建证书的配置集包括的已撤销证书。*
- *由原因代码的 CRL 决定 基于撤销原因代码中包含的已撤销证书。*

3. *为每个发布点配置 CRL。详情请查看 第 7.3.2 节 “为每个发行点配置 CRL”。*
4. *设置为发出点配置的 CRL 扩展。详情请查看 第 7.3.3 节 “设置 CRL 扩展”。*
5. *通过为该发出点、DeltaCRLIndicator 或 CRLumber 启用扩展来为发布点设置 delta CRL。*

6. 设置 CRLDistributionPoint 扩展，以包含有关发布点的信息。

7. 将发布 CRL 设置为文件、LDAP 目录或 OCSP 响应程序。有关设置发布的详情，请参阅 第 8 章 发布证书和 CRL。

7.3.1. 配置发行得分

发出点会定义哪些证书包括在新的 CRL 中。默认情况下，为 master CRL 创建一个 master CRL 发出点，其中包含证书管理器的所有已撤销证书列表。

要创建新发布点，请执行以下操作：

1. 打开 CertificateCertificate Systemnbsp;System Console。

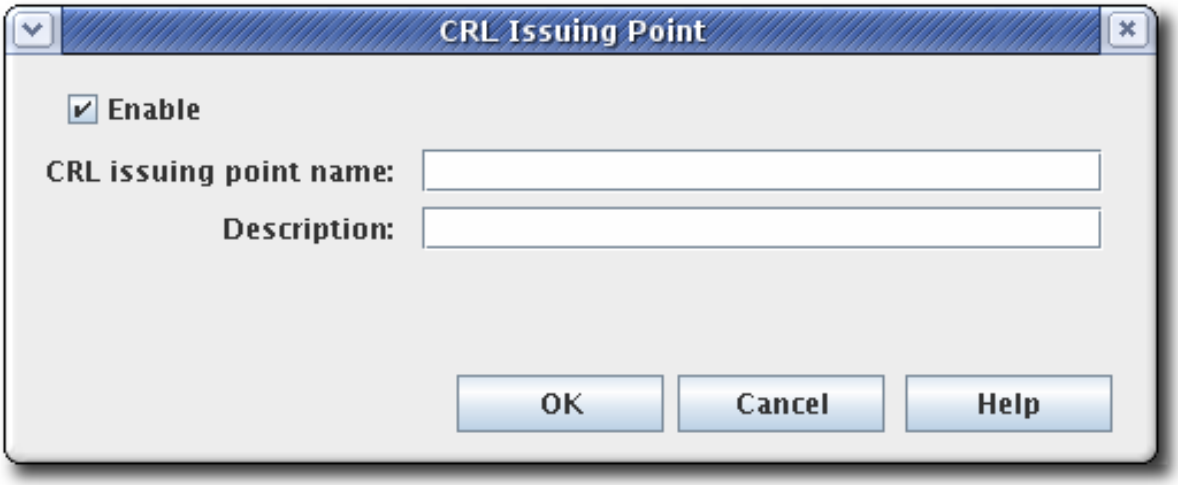
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧导航菜单中展开 Certificate Manager。然后选择 CRLsuing Points。

3. 要编辑发布点，请选择发布点，然后单击 编辑。唯一可以编辑的参数是发出点的名称，以及发出点是否启用还是禁用。

要添加发布点，请单击 Add。CRL Issuing Point Editor 窗口将打开。

图 7.2. CRL Issuing Point Editor



注意

如果某些字段没有足够大来读取内容，请拖动该图标中的一个窗口。

填写以下字段：

- 启用。如果选择，启用发出点；取消选择"禁用"。
- CRL 签发者名称。为发出点指定名称；不允许使用空格。
- 描述。描述发布点。

4. 点击 OK。

要查看并配置新的发布点，请关闭 CA 控制台，然后再次打开控制台。新的发布点列在导航树中的 CRLsuing Points 条目下。

为新的发布点配置 CRL，并设置任何 CRL 扩展来与 CRL 搭配使用。有关配置发布点的详情，请参阅 第 7.3.2 节 “为每个发行点配置 CRL”。有关设置 CRL 扩展的详情，请参阅 第 7.3.3 节 “设置 CRL 扩展”。创建的所有 CRL 会显示在代理服务页面的 Update Revocation List 页面中。

7.3.2. 为每个发行点配置 CRL

在发出点上，都可为 CRL 生成间隔、CRL 扩展和签名算法配置信息。必须为每个发出点配置 CRL。

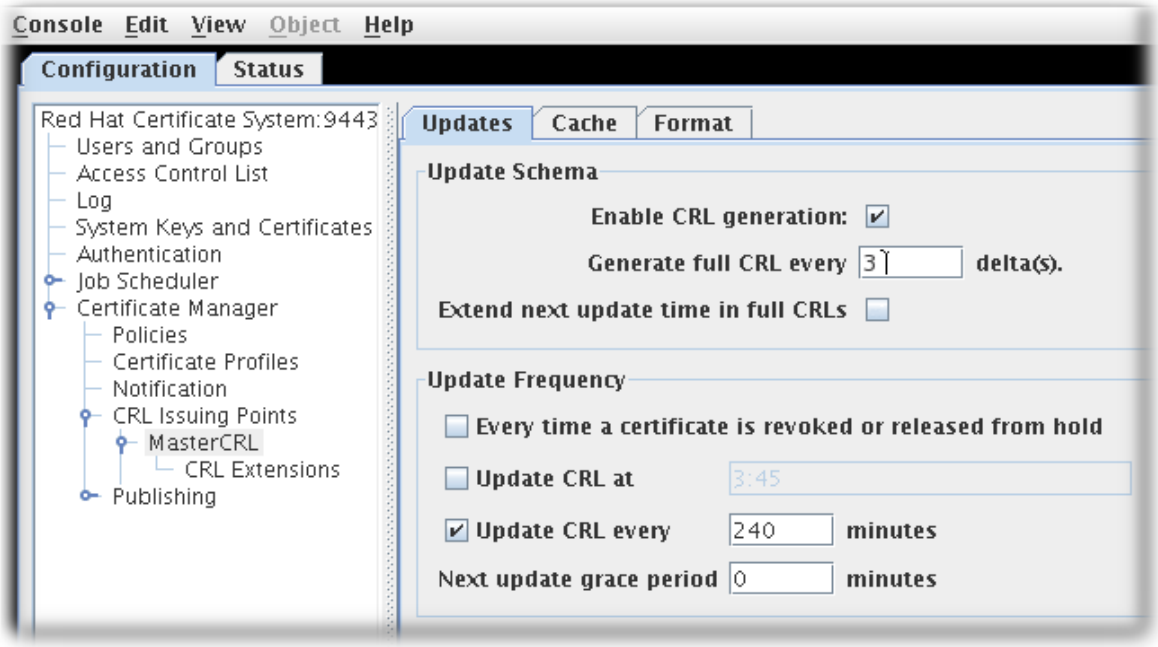
1. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在导航树中，选择 Certificate Manager，然后选择 CRL Issuing Points。

3. 选择 Issuing Points 条目下方的发布点名称。

4. 通过提供发布点的 Update 选项卡中的信息，配置 CRL 如何和更新的频率。此选项卡有两个部分，即 Update Schema 和 Update Frequency。



- Update Schema 部分有以下选项：
  - 启用 CRL 生成。此复选框设置是否为该发出点生成 CRL。
  - 每 # delta(s)生成完整的 CRL。此字段根据更改数量设置如何创建 CRL 的频率。
  -

下次在完整的 CRL 中扩展更新时间。这提供了一个选项，可在生成的 CRLs 中设置 nextUpdate 字段。nextUpdate 参数显示发布下一个 CRL 的日期，无论它是完整的还是 delta CRL。当使用完整和 delta CRL 的组合时，在完整 CRL 中启用下一次 更新时间将使 下一个Update 参数在完整的 CRL 中显示，当下 一个完整的 CRL 时。否则，完整 CRL 中的 nextUpdate 参数会显示何时发出下一个 delta CRL，因为增量是要发布的下一个 CRL。

- Update Frequency 部分在生成 CRL 并发布到目录中时设置不同的间隔。

  - 每次证书被撤销或释放证书时，都会为。这会将证书管理器设置为在每次撤销证书时生成 CRL。证书管理器会在生成 CRL 时尝试向已配置的目录发出 CRL。如果 CRL 较大，则生成 CRL 会消耗大量时间。将证书管理器配置为在每次撤销证书时生成 CRL，可能会使服务器参与大量时间；在此期间，服务器将无法更新接收任何更改的目录。

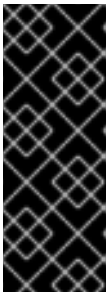
    不建议在标准安装中使用此设置。应该选择这个选项来立即测试撤销，例如测试服务器是否将 CRL 发送到平面文件。

  - 更新位于 的 CRL。此字段在应更新 CRL 时设置每日时间。要多次指定，请输入逗号分隔列表，如 01:50,04:55,06:55。要为多个天输入计划，请输入用逗号分开的列表，在同一天内设置时间，然后是分号隔开的列表来识别不同天数的时间。例如，这会在位于 1:50am、4:55am 和 6:55am，然后在第 2:am、5am 和 5pm 的第 2 天时设置撤销第 1 天：

    ```
    01:50,04:55,06:55;02:00,05:00,17:00
    ```

  - 每一次更新 CRL。这个复选框启用了在字段中设置的间隔生成 CRL。例如，要每天发出 CRL，请选择复选框，然后在此字段中输入 1440。

  - 下次更新宽限期。如果证书管理器以特定频率更新 CRL，可以将服务器配置为在下次更新时有一个宽限期，以允许创建 CRL 并发出它。例如，如果服务器被配置为每 20 分钟更新一次 CRL，则宽限期为 2 分钟，如果 CRL 在 16:00 上更新，则 CRL 会重新更新 16:18。

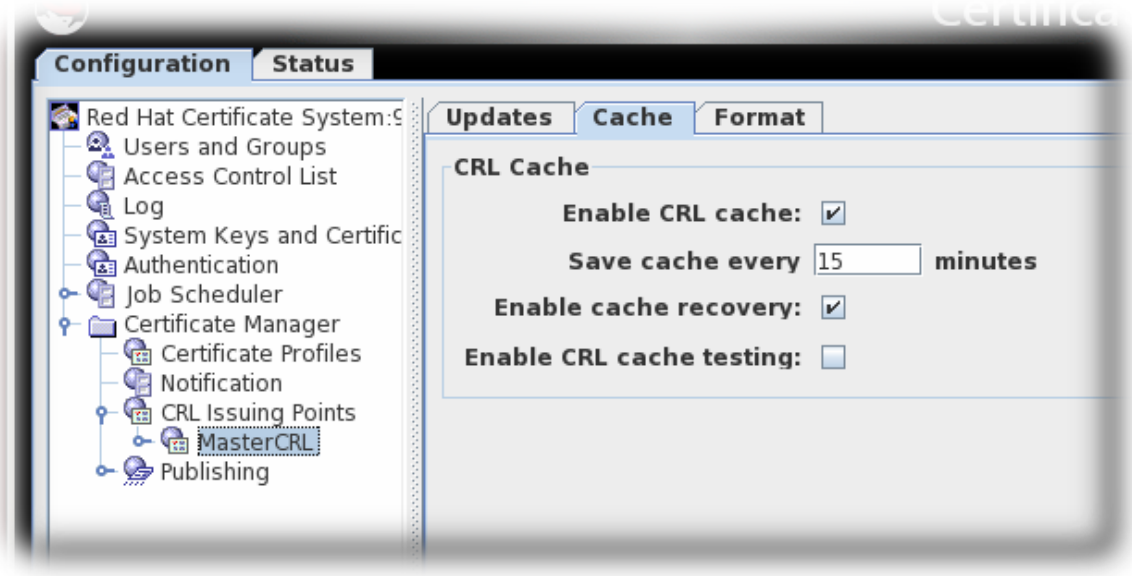


重要

由于一个已知问题，当当前设置 full 和 delta Certificate Revocation List 计划时，每次从 hold 选项撤销或释放证书时，更新 CRL 也要求您填充两个 宽限期 设置。因此，要选择此选项，首先需要选择 更新 CRL 每一个 选项，并为 下一步更新宽限期 # 分钟 输入数字。

5. Cache 选项卡设置是否启用缓存以及缓存频率。

图 7.3. CRL Cache 选项卡



- 启用 CRL 缓存。这个复选框启用了缓存，用于创建 delta CRL。如果禁用了缓存，则不会创建 delta CRL。有关缓存的详情请参考 第 7.1 节 “关于撤销证书”。

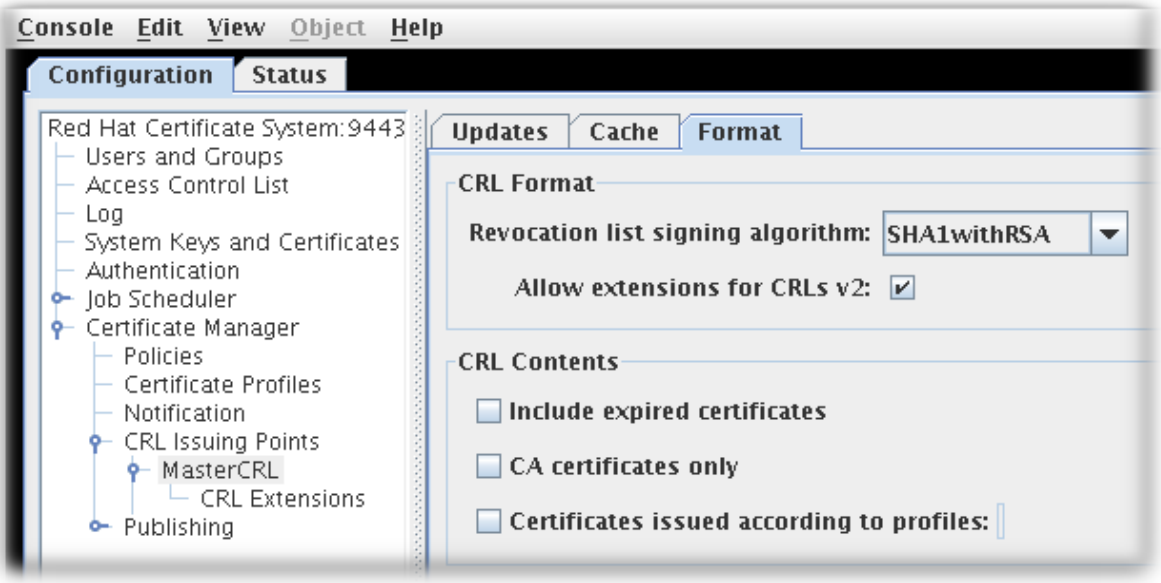
- 更新每个缓存。此字段设定将缓存写入内部数据库的频率。设置为 0， 在每次撤销证书时，将缓存写入数据库。

- 启用缓存恢复。这个复选框允许恢复缓存。

- 启用 CRL 缓存测试。此复选框为特定 CRL 发出点启用 CRL 性能测试。使用这个选项生成的 CRL 不应在部署的 CA 中使用，因为为测试目的发布的 CRL 包含只为性能测试生成的数据。

6. Format 选项卡设定创建的 CRL 的格式和内容。CRL Format 和 CRL 内容有两个部分。

图 7.4. CRL Format 选项卡



- *CRL Format 部分有两个选项：*

  - *撤销列表签名算法 是一个允许加密 CRL 的密码的下拉列表。*

  - *允许 CRL v2 的扩展是一个复选框，为 发出点启用 CRL v2 扩展。如果启用了此项，请设置 第 7.3.3 节 “设置 CRL 扩展” 中描述的所需 CRL 扩展。*

注意

必须开启扩展来创建 delta CRL。

- *CRL Contents 部分有三个复选框，用于设置要在 CRL 中包含哪些类型的证书：*

  - *包括过期的证书。这包括已撤销已过期的证书。如果启用此项，有关已撤销证书的信息会在证书过期后保留在 CRL 中。如果没有启用此项，则证书过期时会删除关于已撤销证书的信息。*

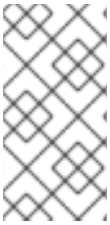
- 仅 CA 证书.这只包括 CRL 中的 CA 证书。选择这个选项会创建一个授权 Revocation List(ARL)，它仅列出已撤销的 CA 证书。

- 根据配置文件发布的证书.这只包括根据列出的配置集发布的证书；指定多个配置集，输入用逗号分开的列表。

7. 点 Save。

8. 此发布点允许扩展，并可配置。详情请查看 第 7.3.3 节 “设置 CRL 扩展”。

7.3.3. 设置 CRL 扩展



注意

只有在为该发出点选择了 Allow extensions for CRLs v2 复选框时，扩展才需要配置为发布点。

创建发布点时，会自动启用三个扩展： CRLReason、InvalidityDate 和 CRLNumber。其他扩展可用，但默认为禁用。这些可以被启用和修改。有关可用 CRL 扩展的更多信息，请参阅 第 B.4.2 节 “标准 X.509 v3 CRL 扩展参考”。

要配置 CRL 扩展，请执行以下操作：

1. 打开 CA 控制台。

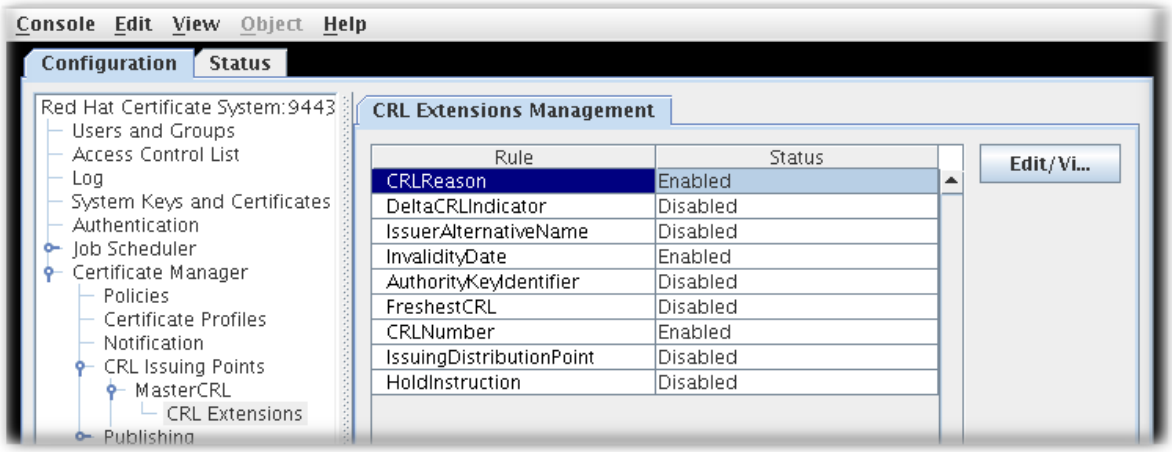
```
pkiconsole https://server.example.com:8443/ca
```

2. 在导航树中，选择 Certificate Manager，然后选择 CRL Issuing Points。

3. 选择发行点条目下的 发布点 名称，并在发布点下选择 CRL Extension 条目。

右窗格显示 CRL Extensions Management 选项卡，它列出了配置的扩展。

图 7.5. CRL 扩展



4. 要修改规则，请选择它，再单击 Edit/View。

5. 大多数扩展有两个选项，启用它们并对其进行设置，无论它们是否至关重要。有些人需要更多信息。提供所有必填值。有关每个扩展以及这些扩展的参数，请参阅 第 B.4.2 节 “标准 X.509 v3 CRL 扩展参考”。

6. 点击 OK。

7. 点 Refresh 查看所有规则的更新状态。

7.3.4. 将 CA 设置为使用其他证书来签名 CRL

有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅在 Red Hat Certificate System Planning、安装和部署指南中的设置 CA 使用其他证书来签名 CRL 部分。

7.3.5. 从缓存生成 CRL

默认情况下，CRL 从 CA 的内部数据库生成。但是，可以收集撤销信息，因为证书会被撤销并保存在内存中。然后，可以使用这个撤销信息从内存中更新 CRL。绕过从内部数据库生成 CRL 所需数据库搜索，显著提高性能。

注意

由于从缓存生成 CRL 的性能增强，请在大多数环境中启用 enableCRLCache 参数。但是，在生产环境中 不应 启用 Enable CRL 缓存测试 参数。

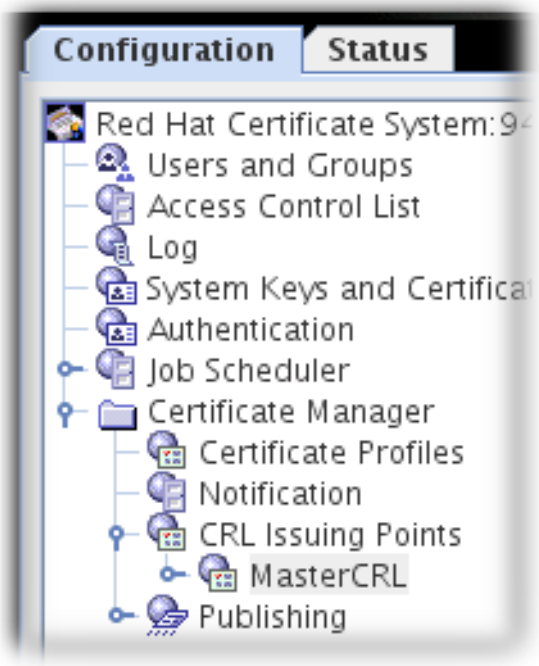
7.3.5.1. 在控制台中配置 CRL Generation from Cache

1. 打开控制台。

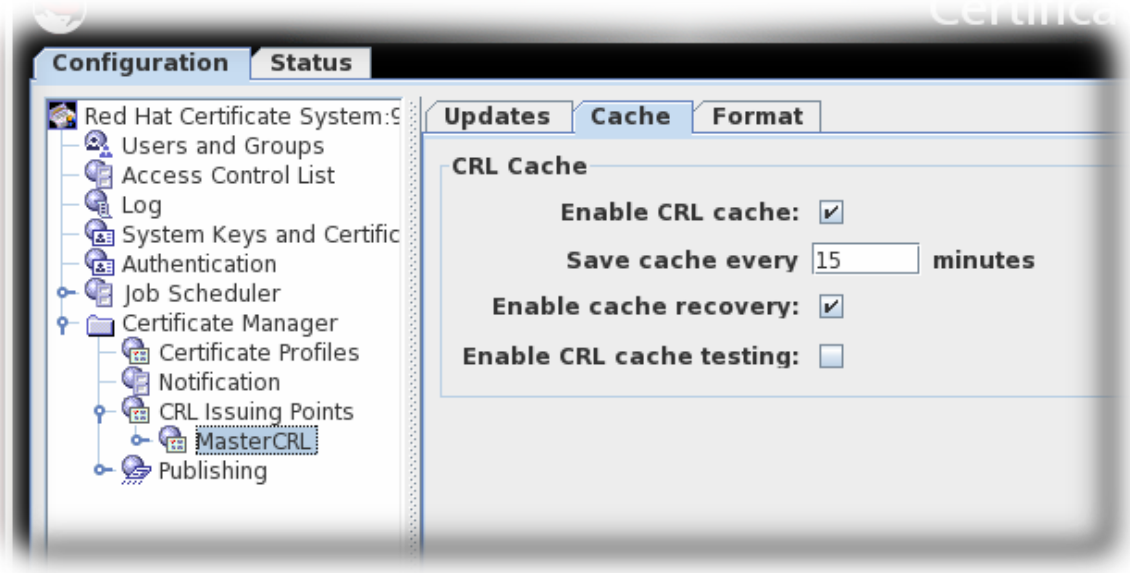
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，展开 Certificate Manager 文件夹和 CRL 签发点 子文件夹。

3. 选择 MasterCRL 节点。



4. 选择 Enable CRL cache。



5. *保存更改。*

7.3.5.2. 在 CS.cfg 中配置 CRL Generation from Cache

有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参考 红帽认证系统规划、安装和部署指南中的 CS.cfg 中的缓存配置 CRL Generation。

7.4. 设置 FULL 和 DELTA CRL SCHEDULES

CRL 定期生成。在 第 7.3.2 节 “为每个发行点配置 CRL” 中的配置中涉及该周期。

根据基于时间的时间表发布 CRL。每次证书被在特定时间或每一天一次撤销一次 CRL 时，可以发布一次 CRL。

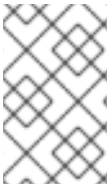
基于时间的 CRL 生成计划适用于生成的每个 CRL。有两种类型的 CRL，完整的 CRL 和 delta CRL。完整的 CRL 拥有每个已撤销的证书，而 delta CRL 仅包含自上次 CRL（增量或完整）被撤销的证书。

默认情况下，按照调度的每个指定间隔生成完整的 CRL。通过生成 增量 CRL，可以在生成完整 CRL 之间耗尽时间。在 CRL 模式 中配置生成间隔，它会设置生成 delta 和完整 CRL 的方案。

如果间隔设为 3，例如，生成的第一个 CRL 将是 full 和 delta CRL，那么下一个生成的更新仅是 delta CRL，则第四个间隔同时是 full 和 delta CRL。换句话说，每个第三代间隔都有完整的 CRL 和

delta CRL。

```
Interval   1, 2, 3, 4, 5, 6, 7 ...
Full CRL   1        4        7 ...
Delta CRL  1, 2, 3, 4, 5, 6, 7 ...
```



注意

除了完整的 CRL 外，要生成 delta CRL，必须启用 CRL 缓存。

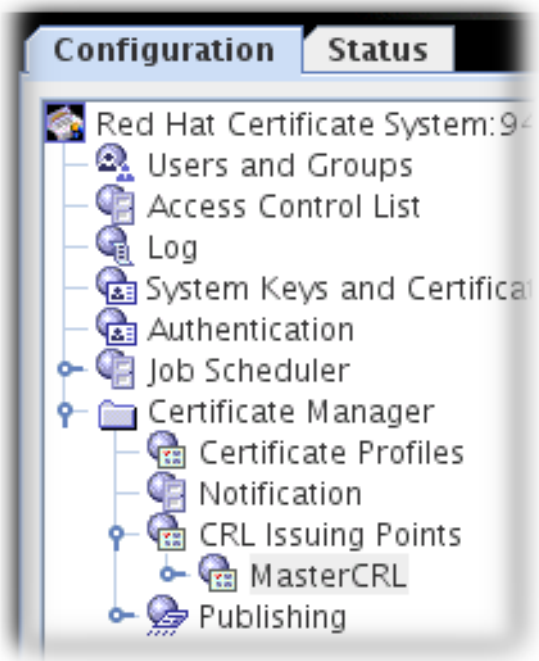
7.4.1. 在控制台中配置 CRL 更新间隔

1. 打开控制台。

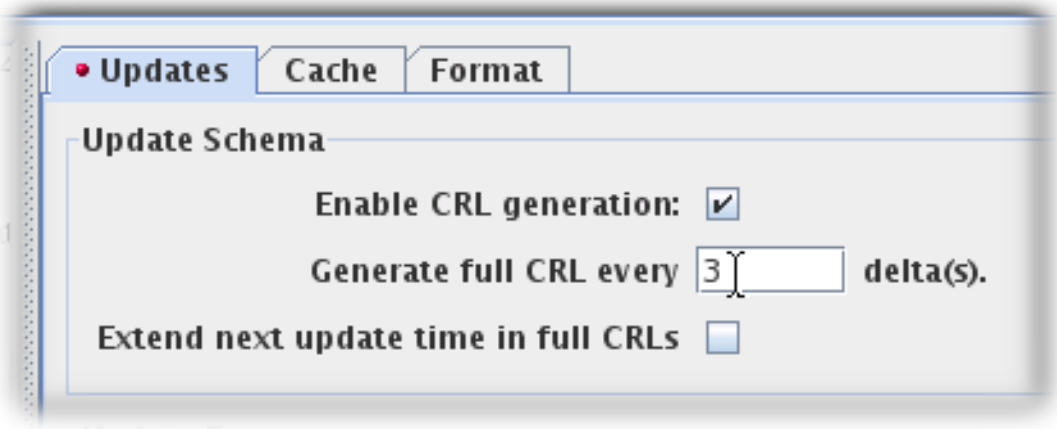
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 **Configuration** 选项卡中，展开 **Certificate Manager** 文件夹和 **CRL** 签发点 子文件夹。

3. 选择 **MasterCRL** 节点。



4. 在 **Generate full CRL every # delta(s)** 字段中输入所需的间隔。

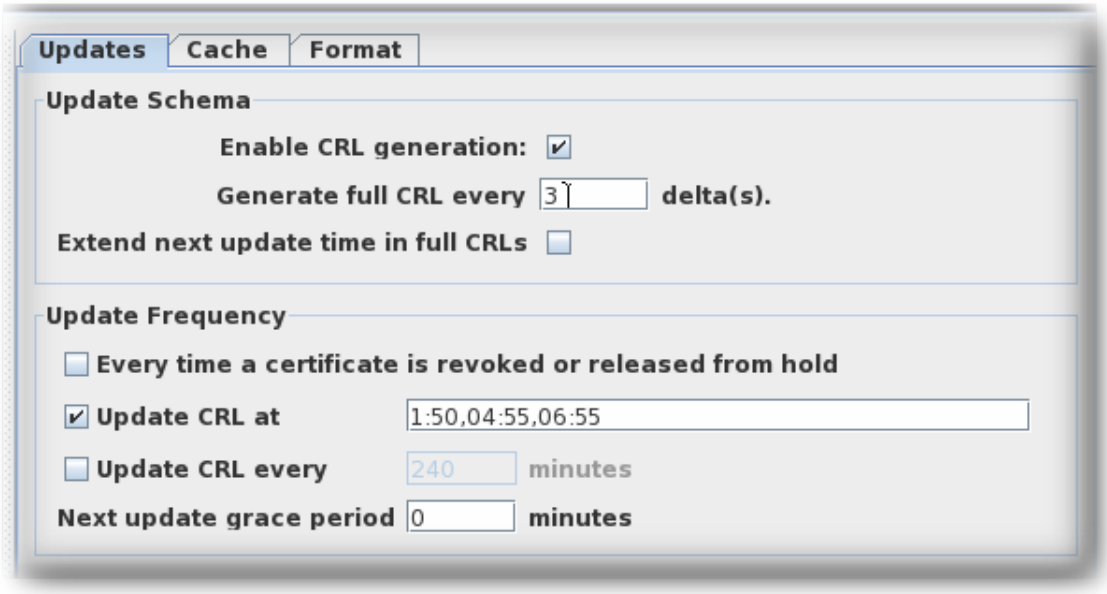


5. 通过指定证书重新调用、循环间隔或设置更新所需的时间来设置更新频率：

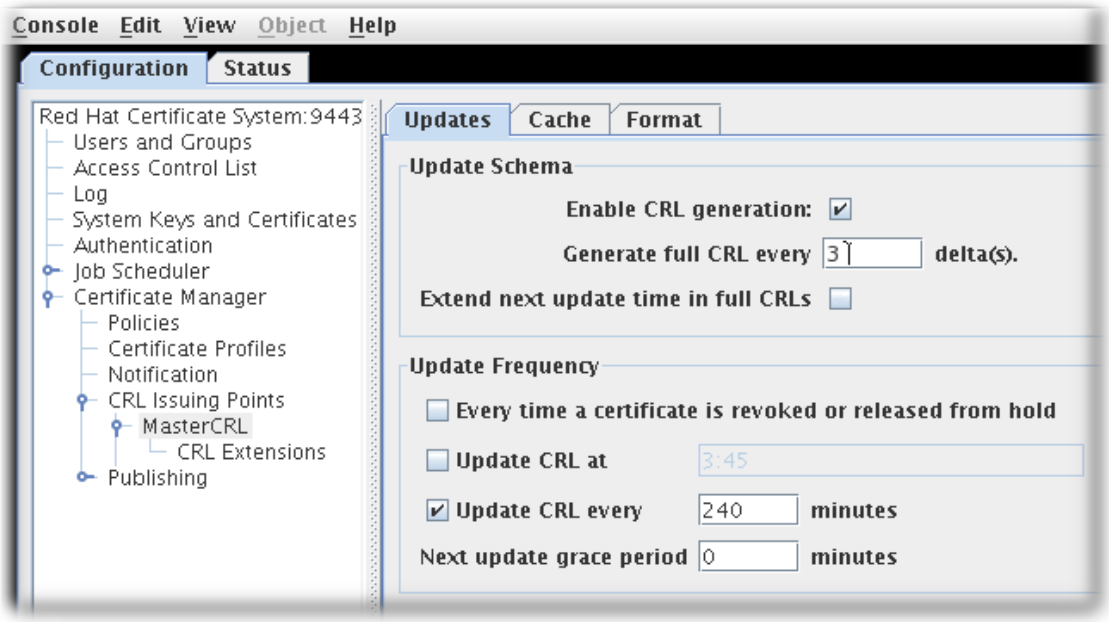
- 每次证书被撤销或释放证书时，选择更新 CRL。每次从 hold 选项撤销或释放证书时，更新 CRL 还需要填写两个 Grace period 设置。这是一个已知问题，错误会在 Red Hat Bugzilla 中跟踪。

- 每次证书被撤销或释放证书时，选择更新 CRL。

- 选中 Update CRL at 复选框，并输入以逗号分开的特定时间，如 01:50,04:55,06:55。

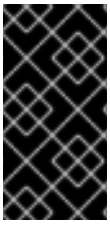


- 选择 更新 CRL 并输入所需间隔，如 240。



6.

保存更改。



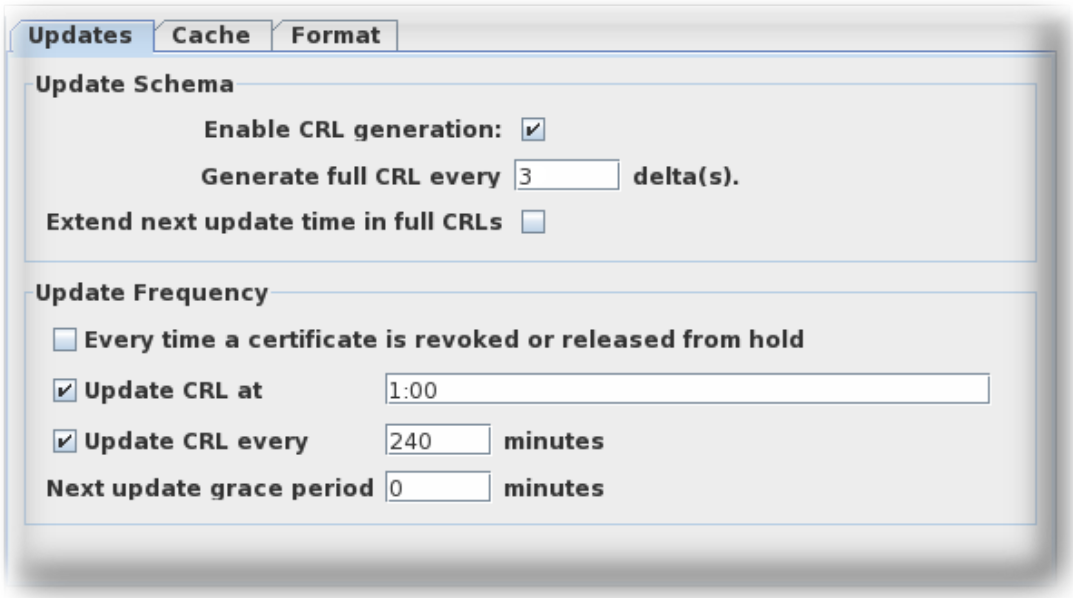
重要

每次从 hold 选项撤销或释放证书时，更新 CRL 还需要填写两个 宽限期 设置。这是一个已知问题，错误会在 Red Hat Bugzilla 中跟踪。

注意

按间隔更新 CRL 时可能会出现调度偏移。通常，偏移会因手动更新和 CA 重启而出现。

要防止调度偏移，请选中 Update CRL at 复选框，然后输入一个值。更新间隔更新将每 24 小时重新同步 更新 CRL 的值。



当按间隔 更新 CRL 时，只接受值的更新 CRL。

7.4.2. 为 CS.cfg 中的 CRL 配置更新间隔

有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅 红帽认证系统规划、安装和部署指南中的 CS.cfg 中的 CRL 配置更新间隔 部分。

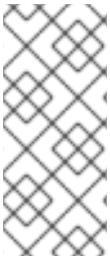
7.4.3. 在多天配置 CRL Generation Schedules

默认情况下，CRL 生成计划时间 24 小时。另外，如果启用了 full 和 delta CRLs，则默认以特定间隔发生完整的 CRL，即每个第三个更新一次或全部 delta CRL。

要在多个天内设置 CRL 生成调度，时间列表使用逗号将同一天内次数隔离，并将分号分隔为分隔符：

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00;02:00,05:00,17:00
```

本例在每天 01:00、03:00 和 18:00 的时间更新 CRL，调度时间为 02:00、05:00 和 17:00。在第三天，周期再次开始。



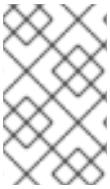
注意

分号表示新日期。使用分号开始列表会导致生成没有 CRL 的初始日。同样，以分号结束的列表将最后一天添加到不生成 CRL 的时间表中。两分号一起会导致每天没有 CRL 生成。

要设置独立于 delta 更新的完整 CRL 更新，则时间值会加上星号，以指示应该发生完整的 CRL 更新：

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00,*23:00;02:00,05:00,21:00,*23:30
```

这个示例在一天 01:00、03:00 和 18:00 中生成 delta CRL 更新，并在 23:00 完整且增量型 CRL 更新。在第二天，增量 CRLs 在 02:00、05:00 和 21:00 进行了更新，并在 23:30 完整且增量型 CRL 更新中。第三天，周期会再次启动。



注意

分号和星号语法在控制台以及手动编辑 CS.cfg 文件时工作。

7.5. 启用撤销检查

撤销检查意味着 CertificateCertificate Systemnbsp;System 子系统验证证书是否有效，并在代理或管理员试图访问实例的安全接口时撤销。这会利用本地 OCSP 服务（CA 的内部 OCSP 服务或单独的 OCSP 响应程序）检查证书的撤销状态。

OCSP 配置包括在 第 7.6 节 “使用在线证书状态协议(OCSP)响应” 中。

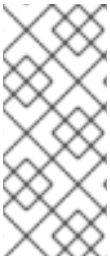
请参阅 红帽认证系统规划、安装和部署指南中的 对 CA 启用自动撤销检查。

请参阅 红帽证书系统规划、安装和部署指南中的启用证书撤销 检查功能。

7.6. 使用在线证书状态协议(OCSP)响应

7.6.1. 设置 OCSP Responder

如果在配置了 Online Certificate Status Manager 时选择了安全域中的 CA，则不需要额外的步骤来配置 OCSP 服务。CA 的 CRL 发布会自动设置，其签名证书会在在线证书状态管理器的证书数据库中自动添加和信任。但是，如果选择了非安全域 CA，必须在配置在线证书状态管理器后手动配置 OCSP 服务。



注意

在配置了 OCSP Manager 时，OCSP Manager 将自动信任 OCSP Manager 中的每个 CA。在 CA 面板中配置的 CA 证书链中的每个 CA 都由 OCSP Manager 自动信任。安全域中的其他 CA，但不能在证书链中被手动信任。

要为安全域以外的证书管理器设置在线证书状态管理器：

1. 为每个将发布到 OCSP 响应器的 CA 配置 CRL。

2. 启用发布、设置发布程序，并在 OCSP 服务将处理每个 CA 中设置发布规则(第 8 章 发布证书和 CRL)。如果证书管理器发布到 LDAP 目录，并且将在线证书状态管理器设置为从该目录中读取，则不需要这一步。

3. 证书配置集必须被配置为包括授权信息访问扩展，指向证书管理器在哪个位置侦听 OCSP 服务请求(第 7.6.4 节 “启用证书管理器的内部 OCSP 服务”)。

4. 配置 OCSP Responder。

   - 配置 Revocation Info 存储（第 7.6.2.2 节 “配置 Revocation Info Stores: Internal Database” 和 第 7.6.2.3 节 “配置 Revocation Info Stores: LDAP Directory”）。

   - 识别每个发布证书管理器到 OCSP 响应器(第 7.6.2 节 “将 CA 识别到 OCSP Responder”)。

   - 如有必要，配置对对 OCSP 签名证书(第 16.7 节 “更改 CA 证书的信任设置”)的 CA 的信任设置。

5. 在配置这两个子系统后，重启这两个子系统。

6. 验证 CA 是否已正确连接到 OCSP 响应器(第 7.6.2.1 节 “验证证书管理器和在线证书状态管理器连接”)。

7.6.2. 将 CA 识别到 OCSP Responder

在将 CA 配置为将 CRL 发布到在线证书 Status Manager 之前，必须通过将 CA 签名证书存储在在线证书管理器的内部数据库中来识别 CAL。证书管理器使用与此证书关联的密钥对签名 CRL；在线证书状态管理器对已存储的证书验证签名。

注意

如果在配置了在线证书状态管理器时选择了安全域中的 CA，则不需要额外的步骤来配置在线证书状态管理器来识别 CA；在线证书管理器在 Online Certificate Status Manager 的证书数据库中自动添加并信任 CA 签名证书。但是，如果选择了非安全域 CA，必须在配置了在线证书 Status Manager 后手动将 CA 签名证书添加到证书数据库中。

不需要为 CA 导入证书链，它将将其 CRL 发布到在线证书 Status Manager。OCSP 服务只需要证书链的唯一时间是，如果 CA 在发布其 CRL 时通过 SSL/TLS 身份验证连接到在线证书状态管理器。否则，在线证书状态管理器不需要具有完整的证书链。

但是，在线证书状态管理器必须具有对 CRL 签名的证书（CA 签名证书或单独的 CRL 签名证书），在其证书数据库中。OCSP 服务将 CRL 的证书与数据库中的证书（而不是针对证书链）进行比较来验证 CRL。如果 root CA 和其下从属 CAs 将 CRL 发布到在线证书 Status Manager，在线证书 Status Manager 需要两个 CA 的 CA 签名证书。

要导入用于签署 CA 或 CRL 签名证书的 CA 或 CRL 签名证书，并将其发布到在线证书 Status Manager，请执行以下操作：

1. 从 CA 的最终页面获取证书管理器的 base-64 CA 签名证书。

2. 打开 Online Certificate Status Manager 代理页面。URL 的格式是 https://hostname:SSLport/ocsp/agent/ocsp。

3. 在左侧框中，点 Add Certificate Authority。

4. 在表单中，将编码的 CA 签名证书粘贴到标记为 Base 64 编码证书的文本区域中（包括标题和页脚）。

5. 要验证证书是否已成功添加，请在左边框中点击 List Certificate Authorities。

生成的表单应该显示有关新 CA 的信息。此更新、下一次 更新 和 Requests Served Since Startup 字段应显示为零(0)的值。

7.6.2.1. 验证证书管理器和在线证书状态管理器连接

当证书管理器重启时，它会尝试连接到在线证书状态管理器的 SSL/TLS 端口。要验证证书管理器是否确实与在线证书 Status Manager 通信，请检查 此 更新和 下一个更新 字段，该字段应该根据 CA 最后一次通信与在线证书状态管理器进行更新。Requests Served Since Startup 字段应该仍然显示零(0)，因为没有客户端试图查询 OCSP 服务以获取证书撤销状态。

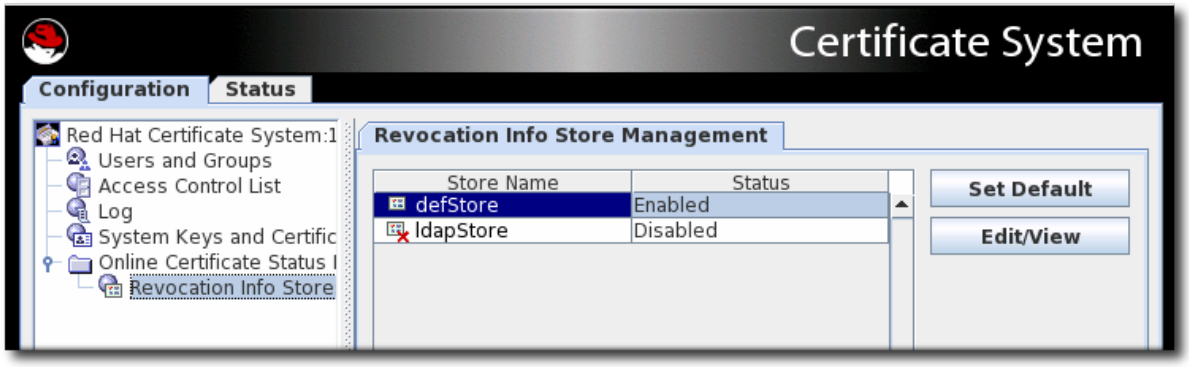
7.6.2.2. 配置 Revocation Info Stores: Internal Database

Online 证书状态管理器将每个证书管理器的 CRL 存储在其内部数据库中，并将其用作 CRL 存储来验证证书的撤销状态。要更改在线证书状态管理器用来将 CRL 存储在其内部数据库中的配置：

1. 打开 Online Certificate Status Manager 控制台。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. 在 Configuration 选项卡中，选择 Online Certificate Status Manager，然后选择 Revocation Info Stores。



右侧窗格显示在线证书状态管理器可以使用的两个存储库；默认情况下，它使用其内部数据库的 CRL。

3. *选择 defStore，再单击 Edit/View。*

4. *编辑 defStore 值。*

- notFoundAsGood.如果问题中的任何 CRL 中找不到证书，则设置 OCSP 服务会返回 GOOD 的 OCSP 响应。如果没有选择此项，则响应是 UNKNOWN，当客户端遇到时，会出现错误消息。

- byName.OCSP Responder 仅支持基本响应类型，其中包括对 OCSP Responder 的 ID。基本响应类型中的 ResponderID 字段由 ocsp.store.defStore.byName 参数的值决定。如果 byName 参数为 true 或缺失，则 OCSP 授权签名证书主题名称将用作 OCSP 响应的 ResponderID 字段。如果 byName 参数为 false，则 OCSP 授权签名证书密钥哈希值将是 OCSP 响应的 ResponderID 字段。

- includeNextUpdate.包括下一次 CRL 更新时间的时间戳。

7.6.2.3. 配置 Revocation Info Stores: LDAP Directory

虽然 OCSP Manager 默认将 CA CRL 存储在其内部数据库中，但您可以将它配置为使用发布到 LDAP 目录的 CRL。

重要

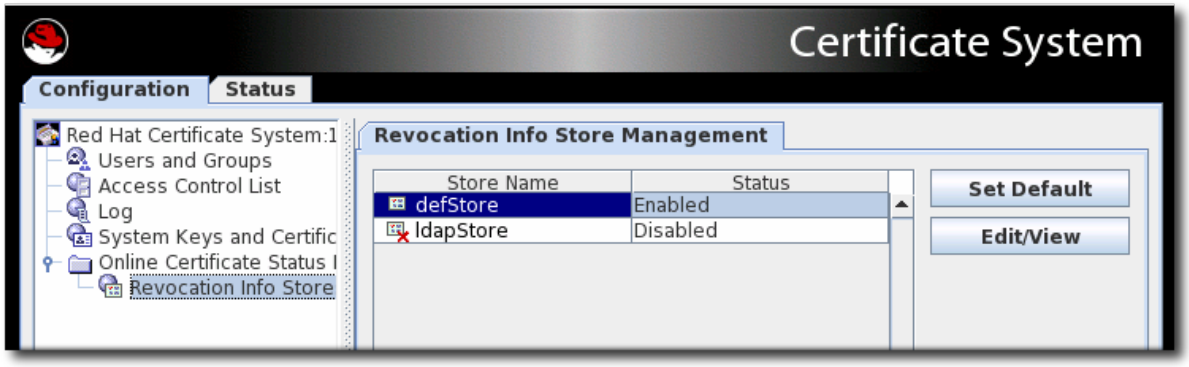
如果启用了 ldapStore 方法，则 OCSP 用户界面不会检查证书状态。

将在线证书状态管理器配置为使用 LDAP 目录：

1. 打开 Online Certificate Status Manager 控制台。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. 在 Configuration 选项卡中，选择 Online Certificate Status Manager，然后选择 Revocation Info Stores。

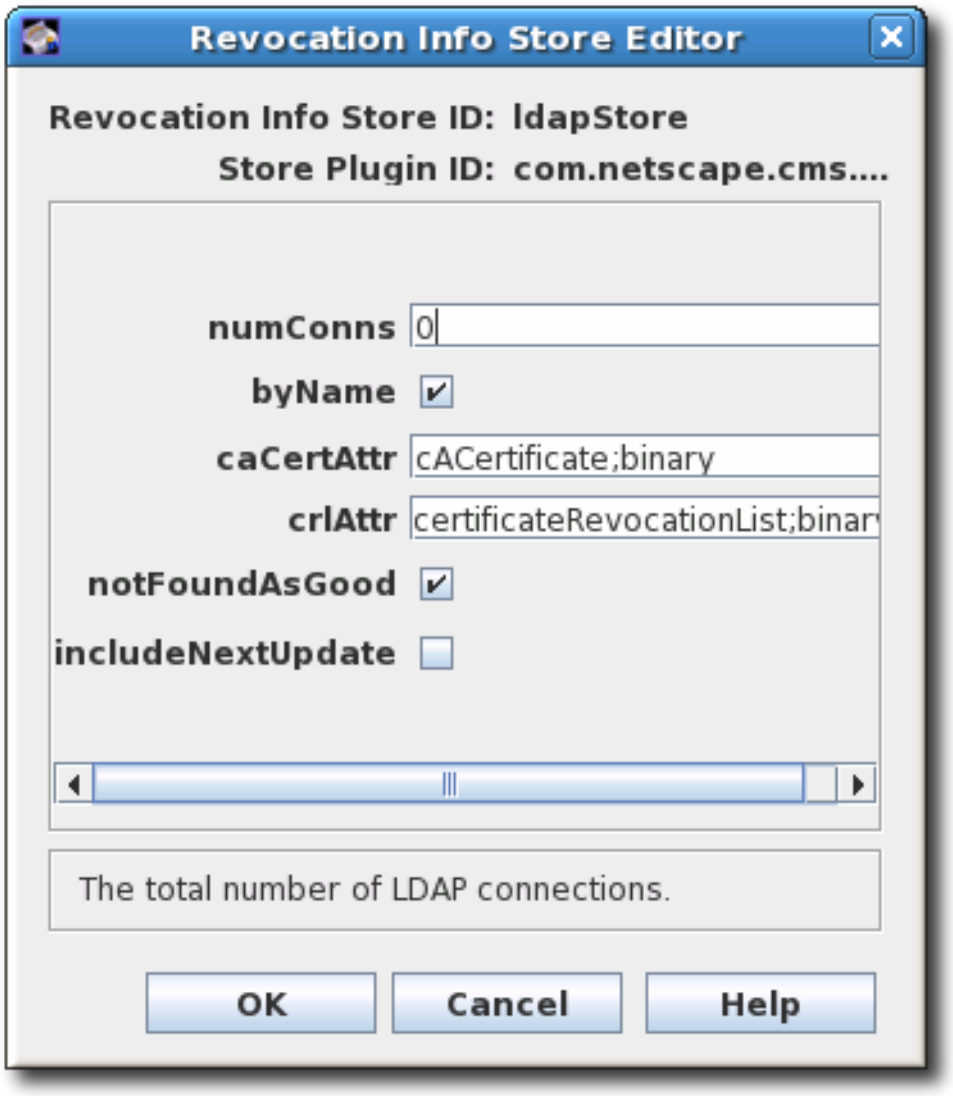


右侧窗格显示在线证书状态管理器可以使用的两个存储库；默认情况下，它使用其内部数据库的 CRL。

3. *要在 LDAP 目录中使用 CRLs，请单击 Set Default 以启用 ldapStore 选项。*

4. *选择 ldapStore，再单击 Edit/View。*

5. *设置 ldapStore 参数。*



- ***numConns.OCSP 服务应检查的 LDAP 目录总数。默认情况下，它被设置为 0。设置此值可显示对应的 主机、端口、baseDN 和 refreshInSec 字段。***

- ***主机.LDAP 目录的完全限定 DNS 主机名。***

- ***端口。LDAP 目录的非 SSL/TLS 端口。***

- ***baseDN.开始搜索 CRL 的 DN。例如： O=example.com。***

- ***refreshInSec.刷新连接的频率。默认值为 86400 秒(daily)。***

- ***caCertAttr.保留默认值 cACertificate;binary，因为它是.它是证书管理器发布其 CA 签名证书的属性。***

- crlAttr.保留默认值 certificateRevocationList;binary，因为它是。它是证书管理器发布 CRL 的属性。

- notFoundAsGood.如果问题中的任何 CRL 中找不到证书，则设置 OCSP 服务会返回 GOOD 的 OCSP 响应。如果没有选择此项，则响应是 UNKNOWN，当客户端遇到时，会出现错误消息。

- byName.OCSP Responder 仅支持基本响应类型，其中包括对 OCSP Responder 的 ID。基本响应类型中的 ResponderID 字段由 ocsp.store.defStore.byName 参数的值决定。如果 byName 参数为 true 或缺失，则 OCSP 授权签名证书主题名称将用作 OCSP 响应的 ResponderID 字段。如果 byName 参数为 false，则 OCSP 授权签名证书密钥哈希值将是 OCSP 响应的 ResponderID 字段。

- includeNextUpdate.Online Certificate Status Manager 可以包括下一个 CRL 更新时间的时间戳。

7.6.2.4. 测试 OCSP 服务设置

通过执行以下操作测试证书管理器是否可以正确服务 OCSP 请求：

1. 在浏览器或客户端中打开撤销检查。
2. 从为 OCSP 服务启用的 CA 请求证书。
3. 批准请求。
4. 将证书下载到浏览器或客户端。
5. 确保 CA 由浏览器或客户端信任。
6. 检查证书管理器的内部 OCSP 服务的状态。

打开 CA 代理服务页面，然后选择 OCSP Services 链接。

7. 测试独立在线证书状态管理器子系统。

打开 Online Certificate Status Manager agent services 页面，然后单击 List Certificate Authorities 链接。

该页面应显示有关证书管理器的信息，以将 CRL 发布到在线证书状态管理器。该页面还总结了自上次启动以来在线证书状态管理器的活动。

8. 撤销证书。

9. 在浏览器或客户端中验证证书。服务器应该返回证书已被撤销。

10. 再次检查证书管理器的 OCSP-service 状态，以验证是否发生了这些问题：

- 浏览器将 OCSP 查询发送到证书管理器。
- 证书管理器向浏览器发送 OCSP 响应。
- 浏览器用于响应验证证书并返回其状态，该浏览器无法验证证书。

11. 再次检查独立的 OCSP 服务子系统，以验证是否发生了这些问题：

- 证书管理器将 CRL 发布到在线证书状态管理器。
- 浏览器将 OCSP 响应发送到在线证书状态管理器。
- 在线证书状态管理器将 OCSP 响应发送到浏览器。

- 浏览器用于响应验证证书并返回其状态，该浏览器无法验证证书。

7.6.3. 为 Bad Serial Numbers 设置响应

OCSP 响应者在确定证书是否有效前检查证书的撤销状态和过期日期。默认情况下，OCSP 不会验证证书上的其他信息。

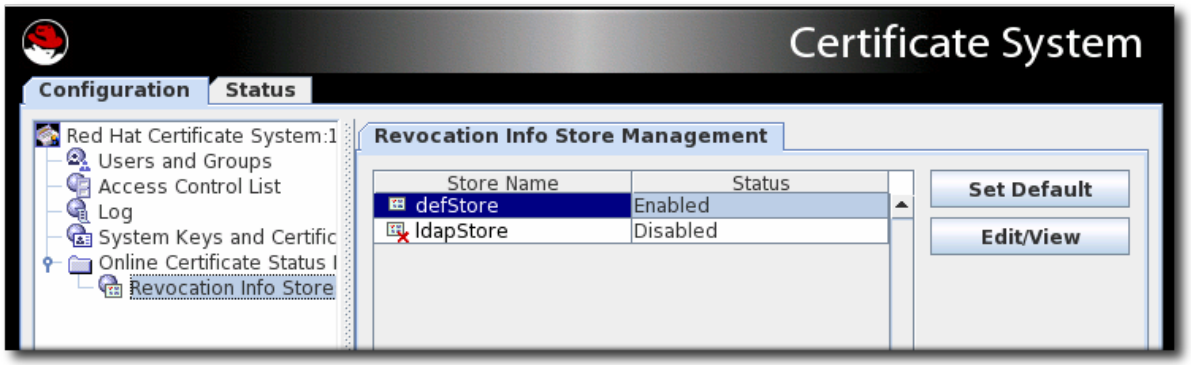
notFoundAsGood 参数设置 OCSP 如何使用无效序列号处理证书。这个参数会被默认启用。这意味着，如果证书存在错误的序列号，但证书无法有效，OCSP 会为证书返回 GOOD 状态。

要让 OCSP 检查并拒绝基于错误序列号和撤销状态的证书，请更改 notFoundAsGood 设置。在这种情况下，OCSP 使用带有错误序列号的证书返回 UNKNOWN 状态。客户端解析为错误并可以相应地做出响应。

1. 打开 Online Certificate Status Manager 控制台。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. 在 Configuration 选项卡中，选择 Online Certificate Status Manager，然后选择 Revocation Info Stores。



3. 选择 defStore，再单击 Edit/View。

4. 编辑 notFoundAsGood 值。选择复选框意味着 OCSP 返回 GOOD 值，即使证书中的序列号不正确。取消选择复选框意味着 OCSP 发送 UNKNOWN 的值，因为客户端可能会将错误作为

错误。

5. 重启 OCSP Manager。

```
]# systemctl restart pki-tomcatd@instance-name.service
```

7.6.4. 启用证书管理器的内部 OCSP 服务

证书管理器中有一个内置 OCSP 服务，它可供 OCSP 兼容客户端用于查询证书管理器，以直接与证书的撤销状态进行查询。安装证书管理器后，会发出 OCSP 签名证书，默认启用 OCSP 服务。此 OCSP 签名证书用于签署对 OCSP 服务请求的所有响应。由于内部 OCSP 服务检查存储在证书管理器内部数据库中的证书状态，因此发布不必配置为使用此服务。

客户端可以通过证书管理器的非 SSL/TLS 端点端口查询 OCSP 服务。查询证书的撤销状态时，证书管理器将搜索其内部数据库以获取证书，检查其状态，并响应客户端。由于证书管理器具有其发布的所有证书的实时状态，因此这种撤销检查方法是最准确的。

每个 CA 内置 OCSP 服务在安装时打开。但是，为了使用此服务，CA 需要使用授权信息访问扩展发布证书。

1. 进入 CA 的最终用户页面。例如：

```
https://server.example.com:8443/ca/ee/ca
```

2. 查找 CA 签名证书。

3. 在证书中查找 Authority Info Access 扩展，并记录 Location URIName 值，如 https://server.example.com:8443/ca/ocsp。

4. 更新注册配置集，以启用授权信息访问扩展，并将 Location 参数设置为证书管理器的 URI。有关编辑证书配置集的详情请参考 第 3.2 节 “设置证书配置集”。

5. 重启 CA 实例。

```
]# systemctl restart pki-tomcatd@instance-name.service
```

注意

要禁用证书管理器的内部 OCSP 服务，请编辑 CA 的 CS.cfg 文件，并将 ca.ocsp 参数的值更改为 false。

```
ca.ocsp=false
```

7.6.5. 使用 OCSPClient 程序提交 OCSP 请求

OCSPClient 程序可用于执行 OCSP 请求。例如：

```
]# OCSPClient -h server.example.com -p 8080 -d /etc/pki/pki-tomcat/alias -c "caSigningCert cert-pki-ca" --serial 2
CertID.serialNumber=2
CertStatus=Good
```

OCSPClient 命令可用于以下命令行选项：

表 7.1. 可用的 OCSPClient 选项

选项	描述
-d *database*	安全数据库位置（默认：当前目录）
-h *hostname*	OCSP 服务器主机名（默认： example.com）
-p *port*	OCSP 服务器端口号（默认： 8080）
-t *path*	OCSP 服务路径（默认：/ocsp/ee/ocsp）
-c *nickname*	CA 证书别名(defaut: CA Signing Certificate)
-n *times*	提交数（默认：1）
--serial *serial_number*	要检查的证书的序列号
--input *input_file*	包含 DER 编码的 OCSP 请求的输入文件
--output *output_file*	保存 DER 编码的 OCSP 响应的输出文件
-v, --verbose	在详细模式下运行
--help	显示帮助信息

7.6.6. 使用 GET 方法提交 OCSP 请求

OCSP 请求可以使用 GET 方法提交到在线证书状态管理器，如 RFC 6960 所述。要通过 GET 提交 OCSP 请求：

1. *为要查询的证书生成 OCSP 请求。例如：*

```
]# openssl ocsp -CAfile ca.pem -issuer issuer.pem -serial serial_number -reqout - | base64

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. *粘贴 Web 浏览器的地址栏中的 URL，返回状态信息。浏览器必须能够处理 OCSP 请求。*

```
https://server.example.com:8443/ocsp/ee/ocsp/MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4
cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy35o7xW5BMzM8FTvyTwCAQE=
```

3. OCSP Manager 使用浏览器可解释的证书状态进行响应。可能的状态是 GOOD、REVOKED 和 UNKNOWN。

或者，使用 curl 等工具从命令行运行 OCSP，以发送请求和 openssl 来解析响应。例如：

1. 为要查询的证书生成 OCSP 请求。例如：

```
]# openssl ocsp -CAfile ca.pem -issuer issuer.pem -serial serial_number -reqout - | base64

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. 使用 curl 连接到 OCSP Manager，以发送 OCSP 请求。

```
curl
https://server.example.com:8443/ocsp/ee/ocsp/MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4
cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy35o7xW5BMzM8FTvyTwCAQE= >
ocspresp.der
```

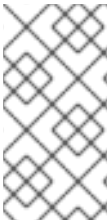
3. 使用 openssl 解析响应：

```
openssl ocsp -respin ocspresp.der -resp_text
```

对于带有授权信息访问扩展的 7.1 CA 发布的证书，需要使用 GET 方法发送到 OCSP，需要创建一个重定向来将请求转发到适当的 URL，如 第 7.6.7 节 “为证书证书系统主任设置重定向：System 7.1 和 Earlier” 所述。

7.6.7. 为证书证书系统主任设置重定向：System 7.1 和 Earlier

OCSP 用户页面的位置，在文件 root /ocsp/ ocsp/ocsp/ 的 URL 中指定，与证书证书系统 nbsp;System 9 或 CertificateCertificate Systemnbsp;System 9 or CertificateCertificate Systemnbsp;System 8.1 的位置与证书证书系统nbsp;System 7.1，只是 .对于要发送到 OCSP 的授权信息访问扩展的 7.1 或更早的 CA 发布的证书，请创建一个重定向以将请求转发到适当的 URL。



注意

设定重定向只需要管理 7.1 或更早的 CA 使用授权信息访问扩展发布的证书。如果证书由后续证书管理器发布，或者不包含授权信息访问扩展，则不需要此配置。

1. 停止 OCSP Responder。

```
]# systemctl stop pki-tomcatd@instance-name.service
```

2. 进入 OCSP 的最终用户 web 应用程序目录。例如：

```
]# cd /var/lib/pki-ocsp/webapps/ocsp
```

3. 更改到 OCSP Web 应用程序目录的 ROOT /WEB-INF/ 目录。例如：

```
]# cd /var/lib/pki-ocsp/webapps/ocsp/ROOT/WEB-INF/
```

4. 在 OCSP 的 web 应用程序目录的 ROOT 文件夹中创建并打开 lib/ 目录。

```
]# mkdir lib
]# cd lib/
```

5. 创建链接回 /usr/share/java/pki/cms.jar JAR 文件的符号链接。例如：

```
]# ln -s /usr/share/java/pki/cms.jar cms.jar
```

6. 移到主 web 应用目录。例如：

```
]# cd /var/lib/pki-ocsp/webapps/ocsp/
```

7. 重命名当前实例(ocsp)目录。例如：

```
]# mv /var/lib/pki-ocsp/webapps/ocsp/ocsp /var/lib/pki-ocsp/webapps/ocsp/ocsp2
```

8. ***更改到原始 ocsp/ 目录中的 WEB-INF/ 目录。例如：***

```
]# cd  /var/lib/pki-ocsp/webapps/ocsp/ocsp/WEB-INF
```

9. ***在原始的 ocsp/WEB-INF/ 目录中，编辑 web.xml 文件并在 eeocspAddCRL 和 csadmin-wizard servlets 之间添加行映射。***

```
  <servlet-mapping>
    <servlet-name>  ocspOCSP  </servlet-name>
    <url-pattern>   /ee/ocsp/*  </url-pattern>
  </servlet-mapping>
```

10. ***在 ROOT 目录中创建并安装 web.xml 文件。例如：***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>

 <display-name>Welcome to Tomcat</display-name>
 <description>
   Welcome to Tomcat
 </description>

 <servlet>
  <servlet-name>ocspProxy</servlet-name>
  <servlet-class>com.netscape.cms.servlet.base.ProxyServlet</servlet-class>
  <init-param>
   <param-name>destContext</param-name>
   <param-value>/ocsp2</param-value>
  </init-param>
  <init-param>
   <param-name>destServlet</param-name>
   <param-value>/ee/ocsp</param-value>
  </init-param>
 </servlet>

 <servlet>
  <servlet-name>ocspOther</servlet-name>
  <servlet-class>com.netscape.cms.servlet.base.ProxyServlet</servlet-class>
  <init-param>
   <param-name>destContext</param-name>
   <param-value>/ocsp2</param-value>
  </init-param>
  <init-param>
   <param-name>srcContext</param-name>
   <param-value>/ocsp</param-value>
  </init-param>
  <init-param>
   <param-name>destServlet</param-name>
   <param-value></param-value>
```

```
    </init-param>
    <init-param>
      <param-name>matchURIStrings</param-name>

<param-value>/ocsp/registry,/ocsp/acl,/ocsp/jobsScheduler,/ocsp/ug,/ocsp/server,/ocsp/log,
          /ocsp/auths,/ocsp/start,/ocsp/ocsp,/ocsp/services,/ocsp/agent,/ocsp/ee,
          /ocsp/admin</param-value>
    </init-param>
    <init-param>
      <param-name>destServletOnNoMatch</param-name>
      <param-value>/ee/ocsp</param-value>
    </init-param>
    <init-param>
      <param-name>appendPathInfoOnNoMatch</param-name>
      <param-value>/ocsp</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>ocspProxy</servlet-name>
    <url-pattern>/ocsp</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>ocspOther</servlet-name>
    <url-pattern>/ocsp/*</url-pattern>
  </servlet-mapping>

</web-app>
```

11. *编辑 **/var/lib/pki-ocsp/conf/context.xml** 文件，更改以下行：*

```
<Context>
 to
<Context crossContext="true" >
```

12. *编辑 **/var/lib/pki-ocsp/webapps/ocsp2/services.template** 文件并更改以下行：*

```
result.recordSet[i].uri);
 to
result.recordSet[i].uri + "/");
```

13. *启动 **OCSP** 实例。*

```
]# systemctl restart pki-tomcatd@instance-name.service
```

部分 III. 管理 CA 服务的其他配置

第 8 章 发布证书和 CRL

Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System 为证书管理器包含一个自定义的发布框架，使证书颁发机构能够发布证书、证书撤销列表(CRL)以及任何受支持存储库的其他证书相关的对象：LDAP 兼容目录、平面文件以及在线验证机构。本章论述了如何配置证书管理器，以将证书和 CRL 发布到文件，以及在线证书状态管理器。

配置发布的一般过程如下：

1. 配置发布到文件、LDAP 目录或 OCSP 响应程序。

   根据所使用的位置数量，可以使用单个发布程序或多个发布者。可以通过证书和 CRL 分割位置，或者分配范围定义，如证书类型。规则通过与发布者关联，确定要发布哪些类型，并告知什么位置。

2. 设置规则以确定将哪些证书发布到位置。激活证书或 CRL 匹配的任何规则，因此同一证书可以通过与基于文件的规则匹配并匹配基于目录的规则，将相同的证书发布到文件以及 LDAP 目录中。

   可以为每个对象类型设置规则： CA 证书、CRL、用户证书和跨对证书。禁用不使用的所有规则。

3. 配置 CRL。必须配置 CRL，然后才能发布。请参阅 第 7 章 撤销证书和发布 CRL。

4. 在设置发布程序、映射程序和规则后发布。启用发布后，服务器开始立即发布。如果没有完全配置发布程序、映射程序和规则，发布可能无法正常工作。

8.1. 关于发布

CertificateCertificate Systemnbsp;System 能够将证书发布到文件或 LDAP 目录，以及将 CRL 发布到文件、LDAP 目录或 OCSP 响应程序。

为了获得额外的灵活性，可以将特定类型的证书或 CRL 发布到单一格式或全部三个格式。例如，CA 证书只能发布到某个目录而不是文件，用户证书可以被发布到一个文件和目录。

注意

OCSP 响应者仅提供有关 CRL 的信息；证书不会发布到 OCSP 响应者。

可以为证书文件和 CRL 文件设置不同的发布位置，以及不同类型的证书文件或不同类型的 CRL 文件的不同发布位置。

同样，可以将不同类型的证书和不同类型的 CRL 发布到目录中的不同位置。例如，公司 West Coast 部门的用户的证书可以在目录的一个分支中发布，而 East Coast division 中的用户的证书可以发布到 目录中的另一个分支。

启用发布后，每次发布证书或 CRL 时，会调用发布系统。证书或 CRL 由规则评估，以查看它是否与规则中设置的类型和 predicate 匹配。类型指定对象是 CRL、CA 证书或任何其他证书。predicate 为被评估的对象类型设置更多条件。例如，它可以指定用户证书，或者指定 West Coast 用户证书。要使用 predicates，需要在发布规则的 predicate 项中输入值，并且对应的值（尽管格式有不同）需要包含在证书或证书请求中以匹配。证书或证书请求中的值可能源自证书中的信息，如证书类型，或者派生自以请求形式放置的隐藏值。如果没有设置 predicate，则该类型的所有证书都将被视为匹配。例如，如果 CRL 设置为类型，则所有 CRL 都匹配规则。

每个匹配的规则都会根据规则中指定的方法和位置发布证书或 CRL。给定证书或 CRL 可以不匹配任何规则、一条规则、多个规则或所有规则。发布系统尝试与针对所有规则发布的每个证书和 CRL 相匹配。

匹配规则时，会根据与该规则关联的发布程序中指定的方法和位置发布证书或 CRL。例如，如果某个规则与向用户发布的所有证书匹配，且该规则有一个发布者，它将发布到位置 /etc/CS/certificates 中的文件，该证书将作为文件发布到该位置。如果另一个规则与用户发布的所有证书匹配，并且该规则有一个发布者，它发布到 LDAP 属性 userCertificate;binary 属性，则会在用户条目的此属性中启用 LDAP 发布时指定的目录。

对于指定要发布到文件的规则，会在被取用的目录中发布证书或 CRL 时创建新文件。

对于指定要发布到 LDAP 目录的规则，证书或 CRL 则在指定的属性中发布到目录中指定的条目。CA 使用任何后续证书或 CRL 来覆盖已发布的证书或 CRL 属性的值。只需放置即可，已经发布的任何现有证书或 CRL 都由下一个证书或 CRL 替代。

对于指定要发布到在线证书状态管理器的规则，此管理器将发布 CRL。证书不会发布到在线证书状态管理器。

对于 LDAP 发布，需要确定用户条目的位置。映射程序用于确定要发布的条目。mappers 可以包含该条目的精确 DN，一些变量关联了可以从证书获取的信息，以创建 DN 或足够信息，用于搜索条目中唯一属性或设置条目的正确 DN。

撤销证书时，服务器使用发布规则从 LDAP 目录或从文件系统中删除对应的证书。

当证书过期时，服务器可以从配置的目录中删除该证书。服务器不会自动执行此操作；服务器必须配置为运行适当的作业。详情请查看 第 12 章 设置自动化任务。

设置发布涉及配置发布程序、映射程序和规则。

8.1.1. publishers

publishers 指定发布证书和 CRL 的位置。当发布到文件时，发布者指定了文件系统发布目录。当发布到 LDAP 目录时，发布程序会指定存储证书或 CRL 的目录中的 属性；mapper 用于确定条目的 DN。对于每个 DN，为推断该 DN 设置一个不同的公式。启用 LDAP 发布时指定 LDAP 目录的位置。将 CRL 发布到 OCSP 响应程序时，发布者指定在线证书 Status Manager 的主机名和 URI。

8.1.2. 映射程序

映射程序 仅在 LDAP 发布中使用。映射程序根据证书或证书请求的信息构建条目的 DN。服务器具有证书的主题名称的信息和证书请求，并需要了解如何使用此信息为该条目创建 DN。映射器提供了一个公式，用于将信息转换为 DN 或某些可在 目录中搜索的唯一信息，以获取该条目的 DN。

8.1.3. 规则

文件、LDAP 和 OCSP 发布的规则会告知服务器是否以及如何发布证书或 CRL。通过为规则设置 type 和 predicate，规则首先定义要发布的内容、与特定特征匹配的证书或 CRL。然后，通过与发布者和 LDAP 发布程序（通过映射程序程序）关联来指定发布方法和位置。

规则可以像 PKI 部署所需一样简单或复杂，并且足够灵活，以适应不同的场景。

8.1.4. 发布到文件

服务器可以将证书和 CRL 发布到平面文件，然后将其导入到任何存储库，如关系数据库。当服务器被配置为将证书和 CRL 发布到文件时，公布的文件为 DER 编码的二进制 blob、base-64 编码的文本 blob

或两者。

- 对于每个服务器问题，它会以 DER-encoded 或 base-64 编码的格式创建一个包含证书的文件。每个文件都命名为 cert-serial_number.der 或 cert-serial_number.b64。serial_number 是文件中包含的证书的序列号。例如，使用序列号 1234 的 DER 编码证书的文件名是 cert-1234.der。

- 每次服务器生成 CRL 时，它会以 DER-encoded 或 base-64 编码的格式创建一个包含新 CRL 的文件。每个文件都命名为 issuing_point_name- this_update.der 或 issuing_point_name- this_update.b64，具体取决于格式。issued_point_name 标识发布 CRL 的 CRL 发出点，this_update 指定从文件中包含的 CRL 独立更新值派生的值。例如，DER 编码 CRL 的文件名以及值 This update: Friday 1:36:00 PST 2020，是 MasterCRL-20200128-153600.der。

8.1.5. OCSP 发布

有两种形式的 CertificateCertificate Systemnbsp;System OCSP 服务，它是证书管理器和在线证书状态管理器的内部服务。内部服务检查证书管理器的内部数据库，以报告证书的状态。内部服务未设置为发布；它使用存储在其内部数据库中的证书来确定证书的状态。Online Certificate Status Manager 检查由证书管理器发送到的 CRL。为每个发送 CRL 的每个位置设置一个发布程序，并为每个发送的 CRL 的规则设置一个规则。

有关两个 OCSP 服务的详情，请参考 第 7.6 节 “使用在线证书状态协议(OCSP)响应”。

8.1.6. LDAP 发布

在 LDAP 中，服务器会使用 LDAP 或 LDAPS 将证书、CRL 和其他与证书相关的对象发布到目录中。它发布的目录的分支称为 发布目录。

- 对于每个证书，它会在用户条目的指定属性中创建一个带有证书的 DER 编码格式的 Blob。证书以 DER 编码的二进制 blob 的形式发布。

- 每次服务器生成 CRL 时，它都会创建一个 blob，它将以 CA 条目的指定属性的 DER 编码格式包含新的 CRL。

服务器可以使用 LDAP 协议或 LDAP 通过 SSL(LDAPS)协议将证书和 CRL 发布到 LDAP 兼容目录，应用可以通过 HTTP 检索证书和 CRL。支持通过 HTTP 检索证书和 CRL，一些浏览器可以从服务器接收定期更新的目录自动导入最新的 CRL。然后，浏览器可以使用 CRL 自动检查所有证书，以确保它们没有被撤销。

要使 LDAP 发布正常工作，用户条目必须位于 LDAP 目录中。

如果服务器和发布目录因某种原因而未同步，则特权用户（管理员和代理）也可以手动启动发布过程。具体说明请查看 第 8.12.2 节 “手动更新目录中的 CRL”。

8.2. 配置发布到文件

配置发布的一般过程涉及设置发布发布程序，以将证书或 CRL 发布到特定位置。根据所使用的位置数量，可以使用单个发布程序或多个发布者。可以通过证书和 CRL 或精细定义（如证书类型）来分割位置。规则通过与发布者关联，确定要发布哪些类型，并告知什么位置。

发布到文件只是将 CRL 或证书发布到给定主机上的文本文件。

必须为每个发布位置创建和配置发布程序；不会自动生成发布发布程序以发布到文件。要将所有文件发布至单一位置，请创建一个发布程序。要发布到不同的位置，请为每个位置创建一个发布程序。位置可以包含对象类型，如用户证书，或者对象类型子集，如 West Coast 用户证书。

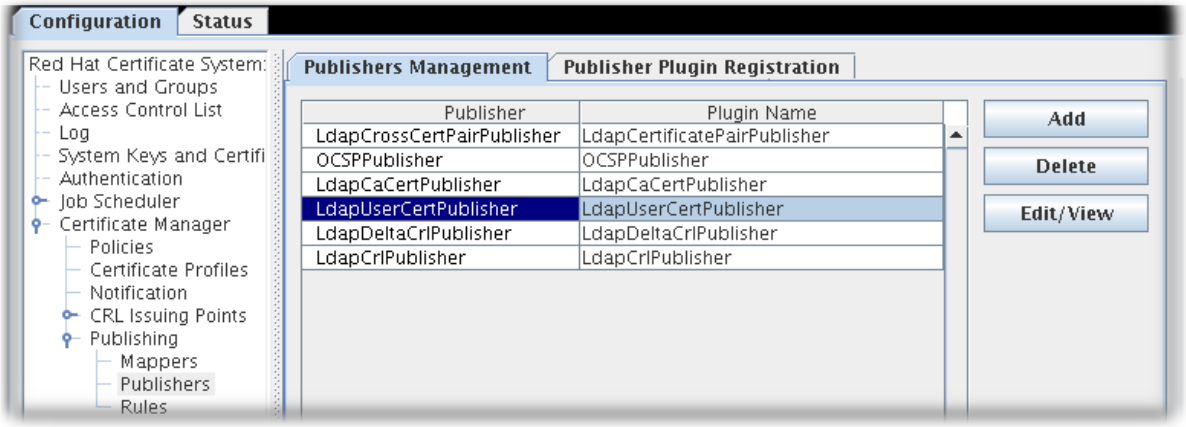
创建发布至文件的发布程序：

1. 登录到证书管理器控制台。

```
pkiconsole https://server.example.com:8443/ca
```

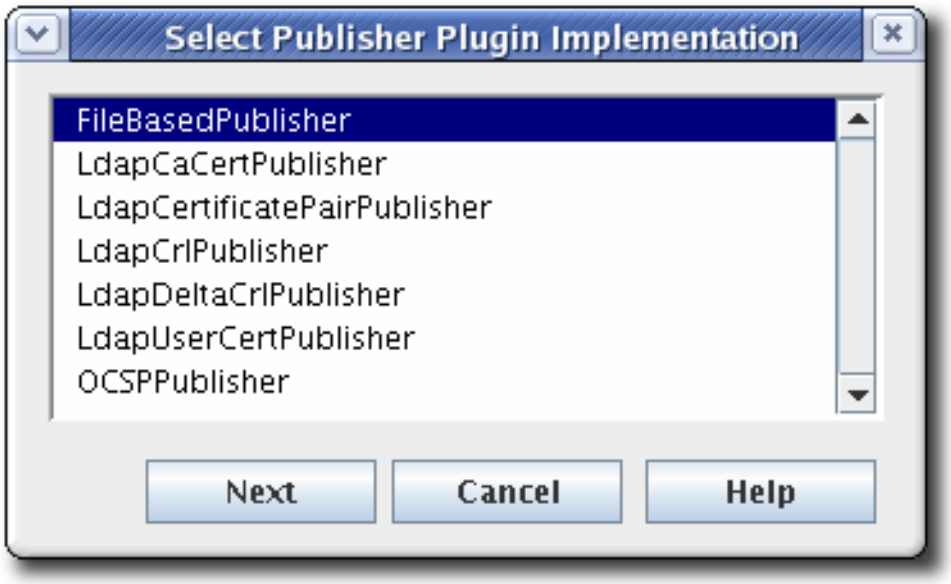
2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布，然后选择 发布软件软件。

publishers Management 选项卡（列出配置的发布程序实例）会在右侧打开。



3.

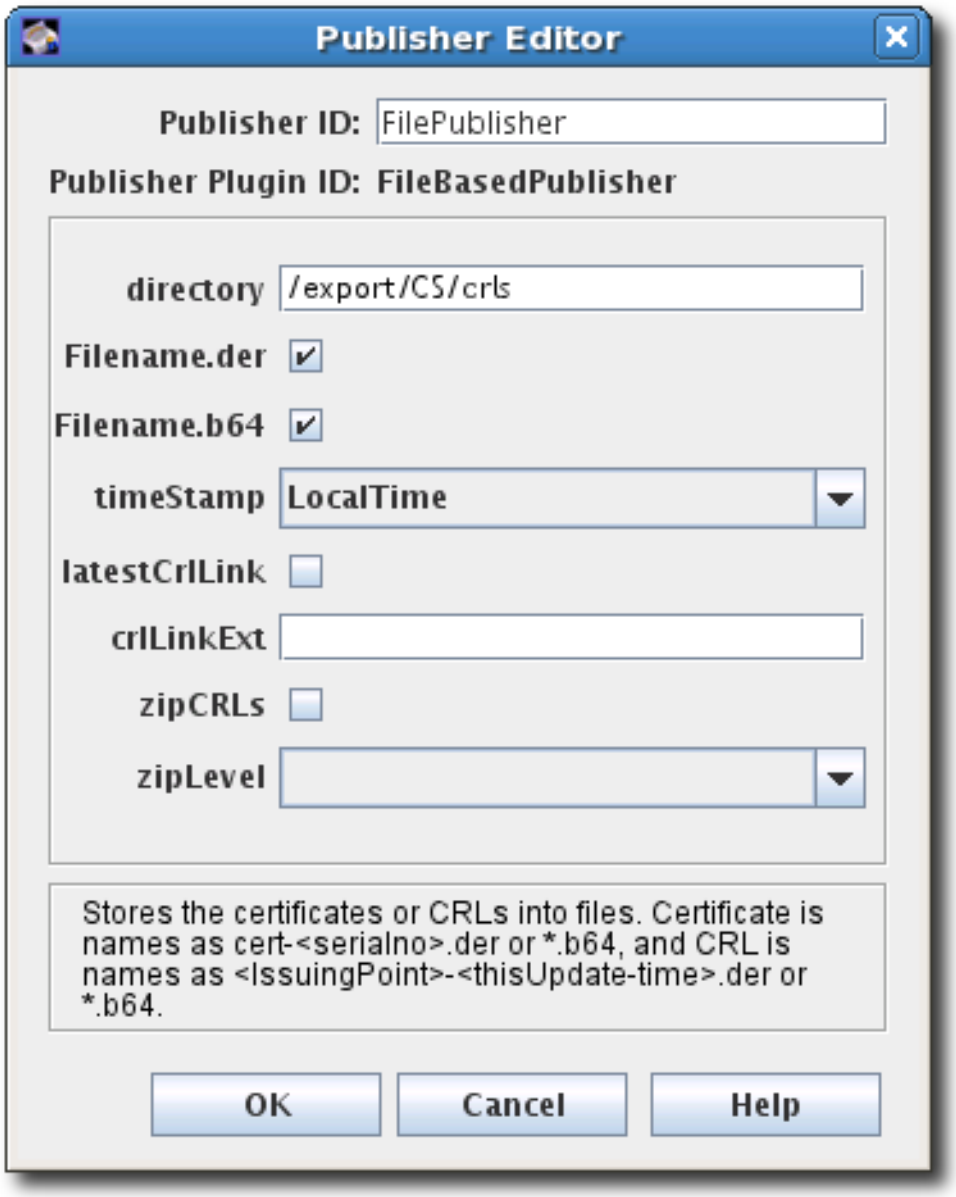
点 Add 打开 Select publisher Plug-in Implementation 窗口，它列出了注册的发布程序模块。



4.

选择 FileBasedPublisher 模块，然后打开编辑器窗口。

这是让证书管理器将证书和 CRL 发布到文件的模块。



5. ***配置发布证书的信息：***

   - ***publisher ID，它是一个字母数字字符串，没有空格，如 PublishCertsToFile***
   - ***证书管理器应发布文件的目录的路径。该路径可以是绝对路径，也可以是相对于 CertificateCertificate Systemnbsp;System 实例目录。例如：/export/CS/certificates。***
   - ***要发布的文件类型，选中 DER 编码文件的复选框、base-64 编码文件或两者。***
   - ***对于 CRL，时间戳的格式。公布的证书在文件名中包括序列号，而 CRL 使用时间戳。***

- 对于 CRL，是否在文件中生成链接以进入最新的 CRL。如果启用，则该链接假定要与扩展一起使用的 CRL 发出的名称将在 crlLinkExt 字段中提供。

- 对于 CRL，是否压缩(zip)CRL 以及要使用的压缩级别。

配置发布程序后，为已发布的证书和 CRL 配置规则，如 第 8.5 节 “创建规则” 所述。

8.3. 配置发布到 OCSP

配置发布的一般过程涉及设置发布发布程序，以将证书或 CRL 发布到特定位置。根据所使用的位置数量，可以使用单个发布程序或多个发布者。可以通过证书和 CRL 或精细定义（如证书类型）来分割位置。规则通过与发布者关联，确定要发布哪些类型，并告知什么位置。

发布到 OCSP Manager 是一种将 CRL 发布到特定位置以进行客户端验证的方法。

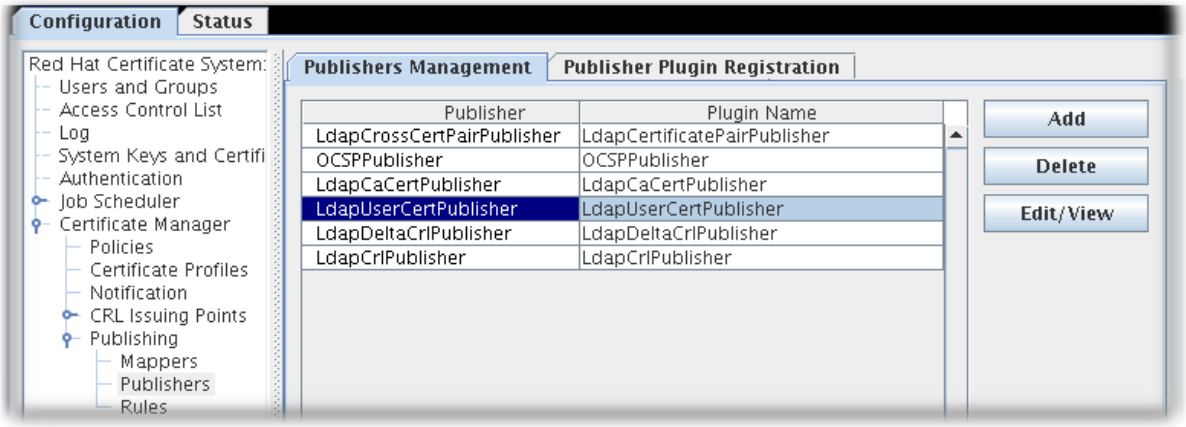
必须为每个发布位置创建和配置发布程序；不会自动创建发布发布程序以发布到 OCSP 响应者。创建一个发布程序，将所有内容发布到单个位置，或为发布 CRL 的每个位置创建一个发布发布发布发布发布器。每个位置都可以包含不同类型的 CRL。

8.3.1. 启用通过客户端身份验证发布到 OCSP

1. 登录到证书管理器控制台。

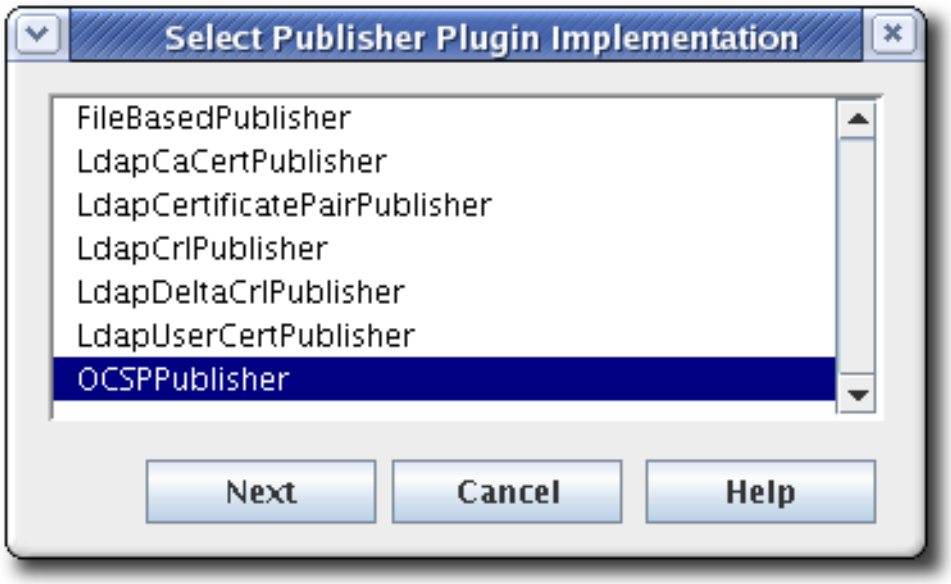
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布，然后选择 发布软件软件。



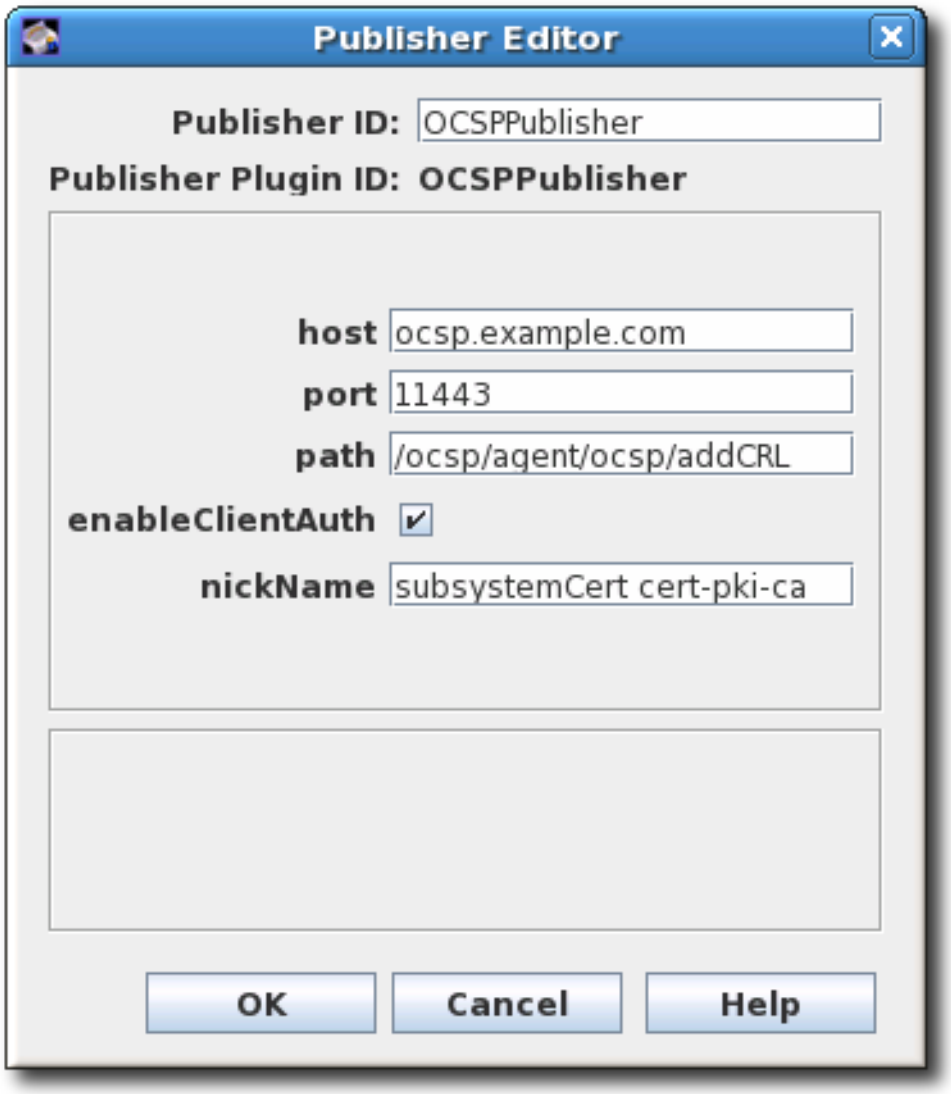
3.

点 Add 打开 Select publisher Plug-in Implementation 窗口，它列出了注册的发布程序模块。



4.

*选择 **OCSPPublisher 模块，然后打开编辑器窗口。这是让证书管理器将 CRL 发布到在线证书状态管理器的 publisher 模块。***



- *publisher ID 必须是字母数字字符串，没有空格，如 PublishCertsToOCSP。*
- *主机 可以是完全限定的域名，如 ocspResponder.example.com，也可以是 IPv4 或 IPv6 地址。*
- *默认路径是要将 CRL 发送到的目录，如 /ocsp/agent/ocsp/addCRL。*
- *如果使用了客户端身份验证（选中了enableClientAuth），则 nickname 字段则提供要用于身份验证的证书的 nickname。此证书必须已存在于 OCSP 安全数据库中，这通常是 CA 子系统证书。*

5. *在 OCSP Manager 上为 CA 创建用户条目。在发送新 CRL 时，用户用于对 OCSP 进行身份验证。需要两个操作：*

- 在 CA 服务器后面命名 OCSP 用户条目，如 CA-hostname-EEport。

- 使用 publisher 配置中指定的任何证书作为 OCSP 用户帐户中的用户证书。这通常是 CA 的子系统证书。

设置子系统用户包括在 第 14.3.2.1 节 “创建用户” 中。

配置发布程序后，为已发布的证书和 CRL 配置规则，如 第 8.5 节 “创建规则” 所述。

8.4. 配置发布到 LDAP 目录

配置发布的一般过程涉及设置发布发布程序，以将证书或 CRL 发布到特定位置。根据所使用的位置数量，可以使用单个发布程序或多个发布者。可以通过证书和 CRL 或精细定义（如证书类型）来分割位置。规则通过与发布者关联，确定要发布哪些类型，并告知什么位置。

配置 LDAP 发布过程与其他发布流程类似，但有额外的步骤来配置目录：

1. 配置将发布到证书的目录服务器。某些属性必须添加到条目中，并且必须配置绑定身份和身份验证方法。

2. 为发布的每种对象类型配置发布程序： CA 证书、跨对证书、CRL 和用户证书。publisher 会声明将对象存储在哪一属性中。默认设置的属性是用于存储每个对象类型的 X.500 标准属性。此属性可以在发布器中更改，但通常不需要更改 LDAP 发布程序。

3. 设置映射程序，使条目的 DN 从证书的主题名称派生出来。这通常不需要为 CA 证书、CRL 和用户证书设置。可为一种证书设置多个映射程序。例如，这可以发布来自位于目录树不同部分公司的两组用户的证书。为每个组创建一个映射程序来指定不同的树分支。

   有关设置映射程序的详情，请参考 第 8.4.3 节 “创建映射程序”。

4. 创建将发布程序连接到映射程序的规则，如 第 8.5 节 “创建规则” 所述。

5.

启用发布，如 第 8.6 节 “启用发布” 所述。

8.4.1. 配置 LDAP 目录

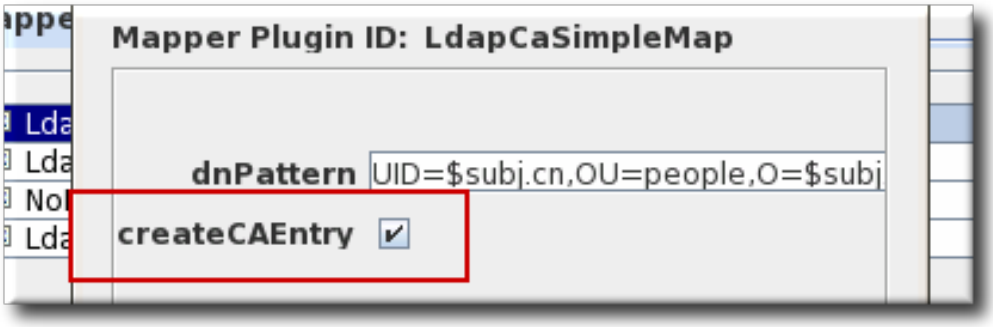
在发布证书和 CRL 之前，必须将 Directory 服务器配置为与发布系统一起使用。这意味着，用户条目必须具有允许它们接收证书信息的属性，而且必须创建条目来代表 CRL。

1.

为 CA 设置条目。要让证书管理器发布其 CA 证书和密钥 CRL，该目录必须包含 CA 的条目。

提示

配置 LDAP 发布后，证书管理器会在 目录中自动创建或转换 CA 的条目。这个选项同时在 CA 和 CRL 映射程序实例中被设置，并默认启用。如果目录限制证书管理器在目录中创建条目，请在这些映射程序实例中关闭此选项，并在 目录中手动添加 CA 的条目。



将 CA 的条目添加到目录时，根据 CA 的 DN 选择条目类型：

- 如果 CA 的 DN 以 cn 组件开头，请为 CA 创建一个新的 人 条目。选择其他类型的条目可能不允许指定 cn 组件。

- 如果 CA 的 DN 从组件开始，请为 CA 创建新的 organizationalunit 条目。

该条目不必处于 pkiCA 或 certificationAuthority 对象类中。证书管理器通过发布其 CA 的签名证书，自动将此条目转换为 pkiCA 或 certificationAuthority 对象类。

注意

pkiCA 对象类在 RFC 4523 中定义，而 Certified Authority 对象类在 (obsolete)RFC 2256 中定义。根据 Directory 服务器使用的 schema 定义，可以接受任一对象类。在某些情况下，两个对象类都可用于同一 CA 条目。

有关创建目录条目的更多信息，请参阅 Red Hat Directory Server 文档。

2.

在 CA 和用户目录条目中添加正确的模式元素。

要让证书管理器将证书和 CRL 发布到某个目录，必须配置有特定属性和对象类。

对象类型	模式	原因
最终用户证书	userCertificate;binary (attribute)	这是证书管理器发布证书的属性。 这是一个多值属性，每个值都是 DER 编码的二进制 X.509 证书。名为 **inetOrgPerson** 的 LDAP 对象类允许此属性。**strongAuthenticationUser** 对象类允许此属性，并可与其他对象类结合使用，以允许将证书与其他对象类一起发布到目录条目。证书管理器不会自动将此对象类添加到相应 Directory 服务器的 schema 表中。 如果它找到的目录对象不允许 **userCertificate;binary** 属性，添加或删除证书会失败。
CA 证书	caCertificate;binary (attribute)	这是证书管理器发布证书的属性。 服务器启动时，证书管理器将自己的 CA 证书发布到自己的 LDAP 目录条目。该条目对应于证书管理器的签发者名称。 这是 **pkiCA** 或 **certificationAuthority** 对象类的必要属性。如果证书管理器可以找到 CA 的目录条目，则证书管理器将此对象类添加到 CA 的目录条目中。

对象类型	模式	原因
CRL	certificateRevocationList;binary (attribute)	这是证书管理器发布 CRL 的属性。 证书管理器将 CRL 发布到自己的 LDAP 目录条目。该条目对应于证书管理器的签发者名称。 这是 **pkiCA** 或 **certificationAuthority** 对象类的属性。属性的值是 DER 编码的二进制 X.509 CRL。CA 的条目必须已经包含 **pkiCA** 或 **certificationAuthority** 对象类，才能将 CRL 发布到该条目。
Delta CRL	deltaRevocationList;binary (attribute)	这是证书管理器发布 delta CRL 的属性。证书管理器将 delta CRL 发布到自己的 LDAP 目录条目，独立于完整的 CRL。delta CRL 条目对应于证书管理器的签发者名称。 此属性属于 **deltaCRL** 或 Certification **Authority-V2** 对象类。属性的值是 DER 编码的二进制 X.509 delta CRL。

3. *为证书管理器设置绑定 DN，以用于访问目录服务器。*

证书管理器用户必须对目录具有读写权限，才能将证书和 CRL 发布到该目录，以便证书管理器可以修改与证书相关的信息的用户条目，以及 CA 的证书和 CRL 相关信息的 CA 条目。

绑定 DN 条目可以是以下任意一种：

- *具有写入权限的现有 DN，如 Directory Manager。*
- *被授予写入访问权限的新用户。条目可以通过证书管理器的 DN 识别，如 cn=testCA、ou=Research Dept、o=Example Corporation、st=California, c=US。*

注意

请仔细考虑为此用户授予什么权限。通过为帐户创建 ACL，可以限制该用户在目录中写入的内容。有关授予证书管理器条目的写入权限的说明，请参阅目录服务器文档。

4. 设置目录验证方法，以便证书管理器如何向 Directory 服务器进行身份验证。有三个选项：基本身份验证（简单用户名和密码）；没有客户端身份验证的 SSL（简单用户名和密码）；以及通过客户端身份验证（基于证书）的 SSL。

有关设置这些与服务器通信的说明，请查看 Red Hat Directory Server 文档。

8.4.2. 配置 LDAP 发布程序

证书管理器创建、配置并启用与 LDAP 发布相关的一组发布程序。默认发布程序（用于 CA 证书、用户证书、CRL 和跨对证书）已遵循 X.500 标准属性来存储证书和 CRL，不需要更改。

表 8.1. LDAP 发布程序

publisher	描述
LdapCaCertPublisher	将 CA 证书发布到 LDAP 目录。
LdapCrlPublisher	将 CRL 发布到 LDAP 目录。
LdapDeltaCrlPublisher	将 delta CRL 发布到 LDAP 目录。
LdapUserCertPublisher	将所有类型的终止证书发布到 LDAP 目录。
LdapCrossCertPairPublisher	将跨签名证书发布到 LDAP 目录。

8.4.3. 创建映射程序

映射程序只与 LDAP 发布一起使用。映射程序定义了证书主题名称和发布证书的目录条目的 DN 之间的关系。证书管理器需要从证书或证书请求中获取条目的 DN，以便它能够决定使用哪个条目。映射映射定义了用户条目的 DN 之间的关系以及证书或其他输入信息的主体名称，以便可以确定条目的确切 DN 并位于 目录中。

配置后，证书管理器会自动创建一组定义最常见的关系的映射程序。默认映射程序在 表 8.2 “默认映射程序” 中列出。

表 8.2. 默认映射程序

mapper	描述
LdapUserCertMap	在 目录中找到用户条目的正确属性，以发布用户证书。
LdapCrlMap	在 目录中找到 CA 条目的正确属性，以发布 CRL。
LdapCaCertMap	在 目录中找到 CA 条目的正确属性，以发布 CA 证书。

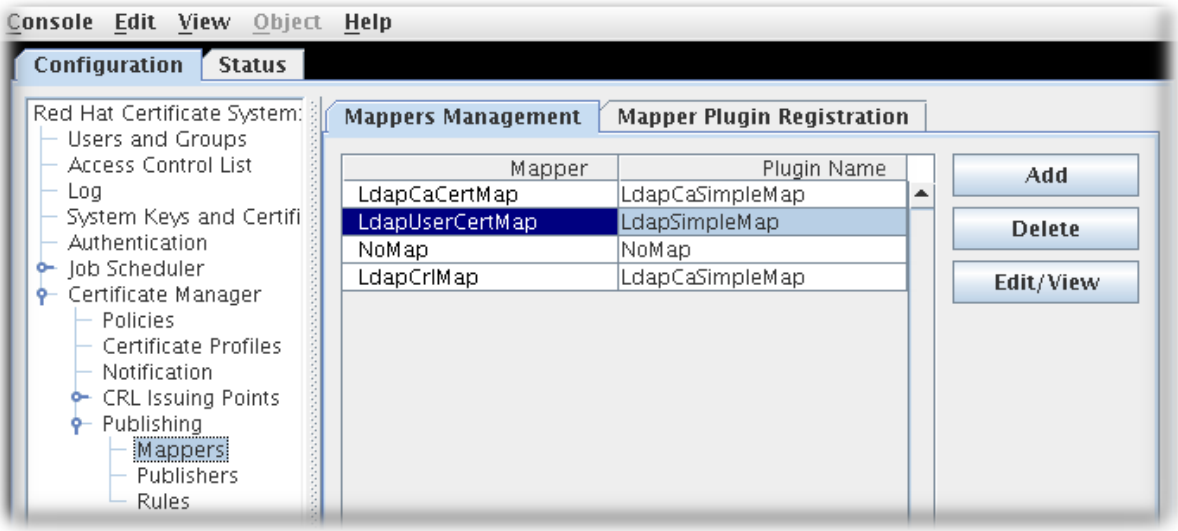
要使用默认映射程序，通过指定 DN 模式以及是否在目录中创建 CA 条目来配置每个宏。要使用其他映射程序，请创建和配置映射程序的实例。更多信息请参阅 第 C.2 节 “映射器插件模块 ”。

1. *登录到证书管理器控制台。*

```
pkiconsole https://server.example.com:8443/ca
```

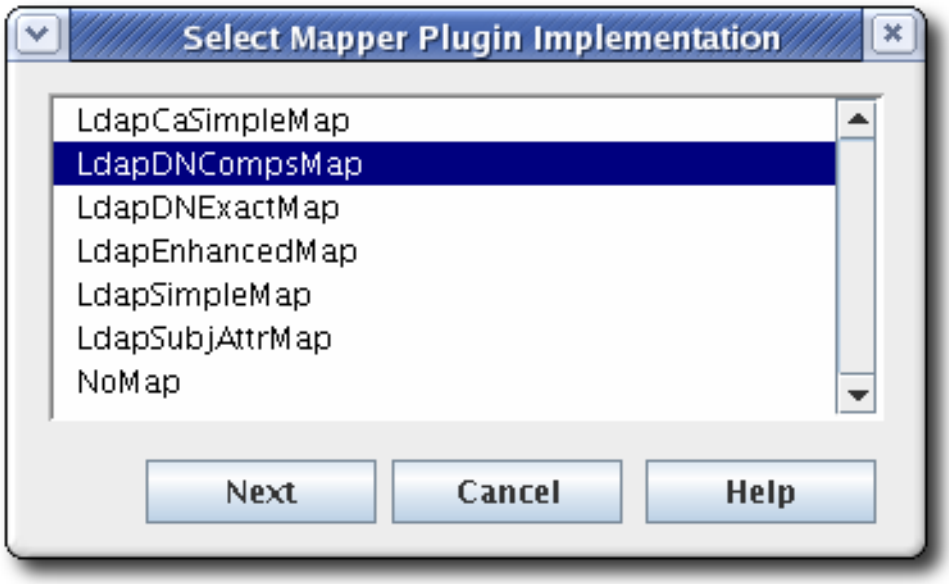
2. *在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布，然后选择 映射程序。*

映射程序管理 标签页（列出配置的映射程序）会在右侧打开。

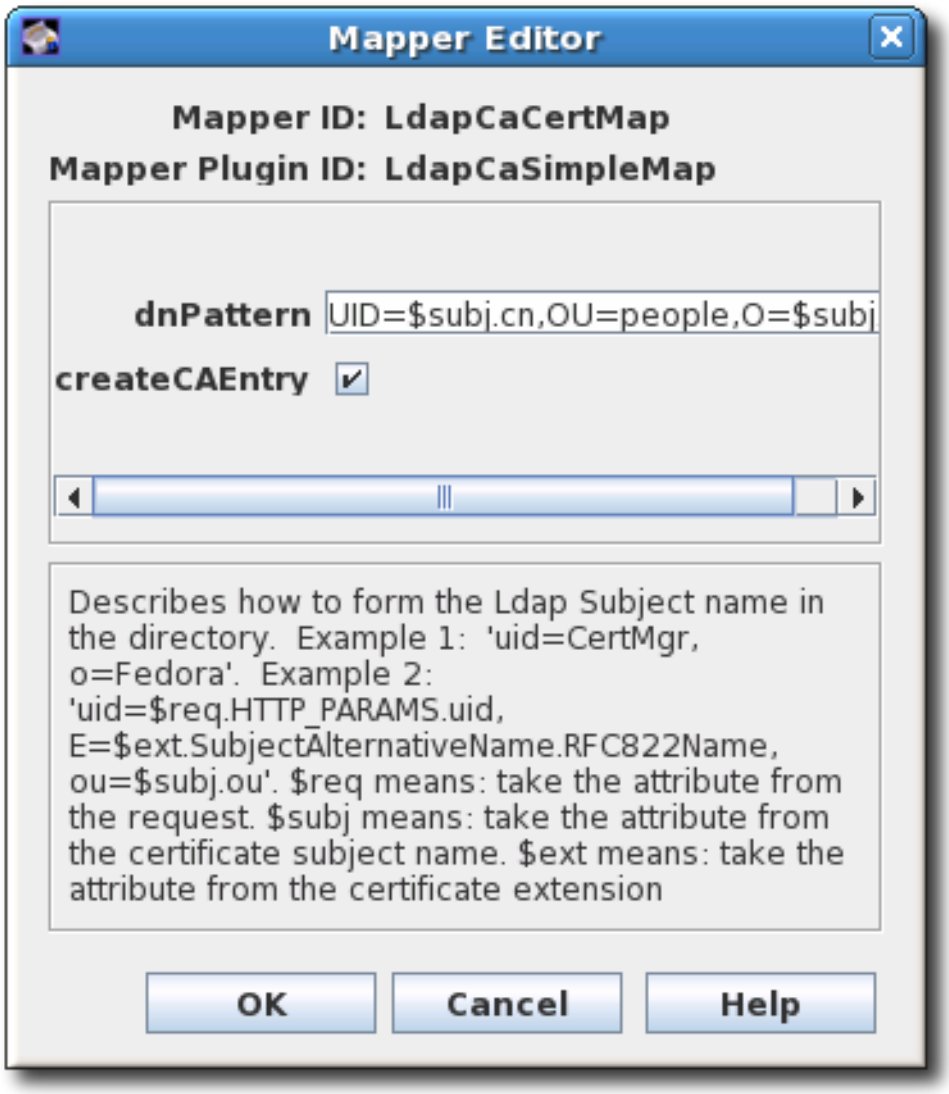


3. *要创建新映射程序实例，请单击 Add。将打开 Select Mapper Plugin Implementation 窗口，它列出了已注册映射程序模块。选择一个模块并进行编辑。有关这些模块的完整信息，请参*

阅 第 C.2 节 “映射器插件模块 ”。



4. *编辑映射程序实例，然后单击确定。*



有关每个映射器的详细信息，请参阅 第 C.2 节 “映射器插件模块 ”。

8.4.4. 完成配置：规则并启用

为 LDAP 发布配置映射程序后，为公布的证书和 CRL 配置规则，如 第 8.5 节 “创建规则” 所述。

配置完成后，启用发布，如 第 8.6 节 “启用发布” 所述。

8.5. 创建规则

规则决定了在什么位置发布哪些证书对象。规则可以独立工作，而不是以 tandem 工作。正在发布的证书或 CRL 与任何规则都匹配。任何与之匹配的规则都会被激活。这样，可以将相同的证书或 CRL 发布到文件、在线证书状态管理器以及通过匹配基于文件的规则、OCSP 规则并匹配基于目录的规则来发布到 LDAP 目录。

可以为每个对象类型设置规则： CA 证书、CRL、用户证书和跨对证书。这些规则对不同类型的证书或不同种类的 CRL 更为详细。

规则首先确定对象是否与规则中设置的类型和 predicate 匹配。发布匹配对象的位置由与该规则关联的发布程序和映射程序决定。

针对证书管理器问题，创建每种类型的规则。

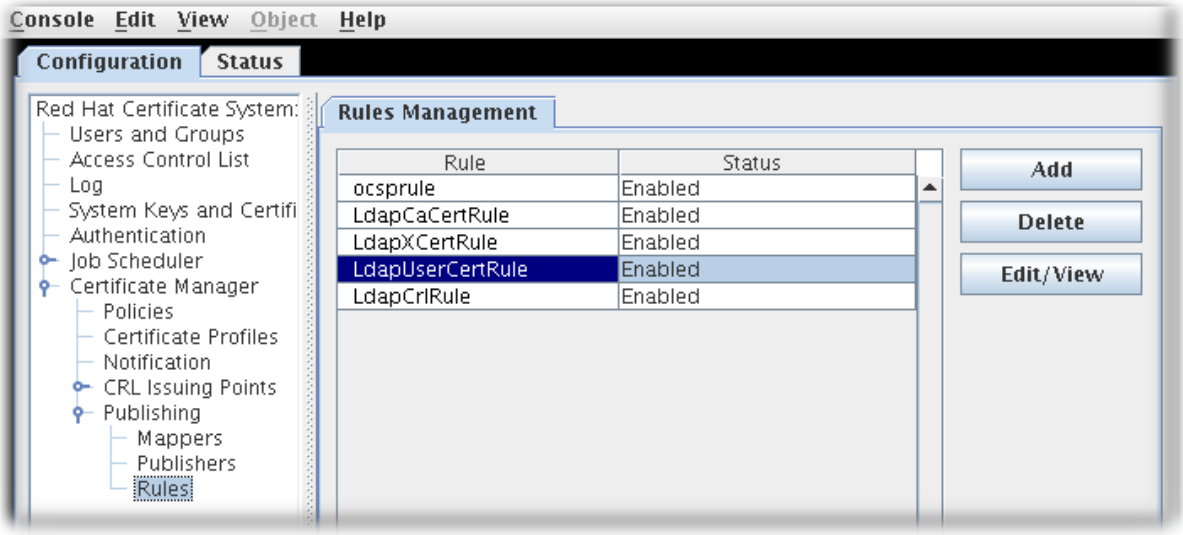
通过执行以下操作修改发布规则：

1. 登录到证书管理器控制台。

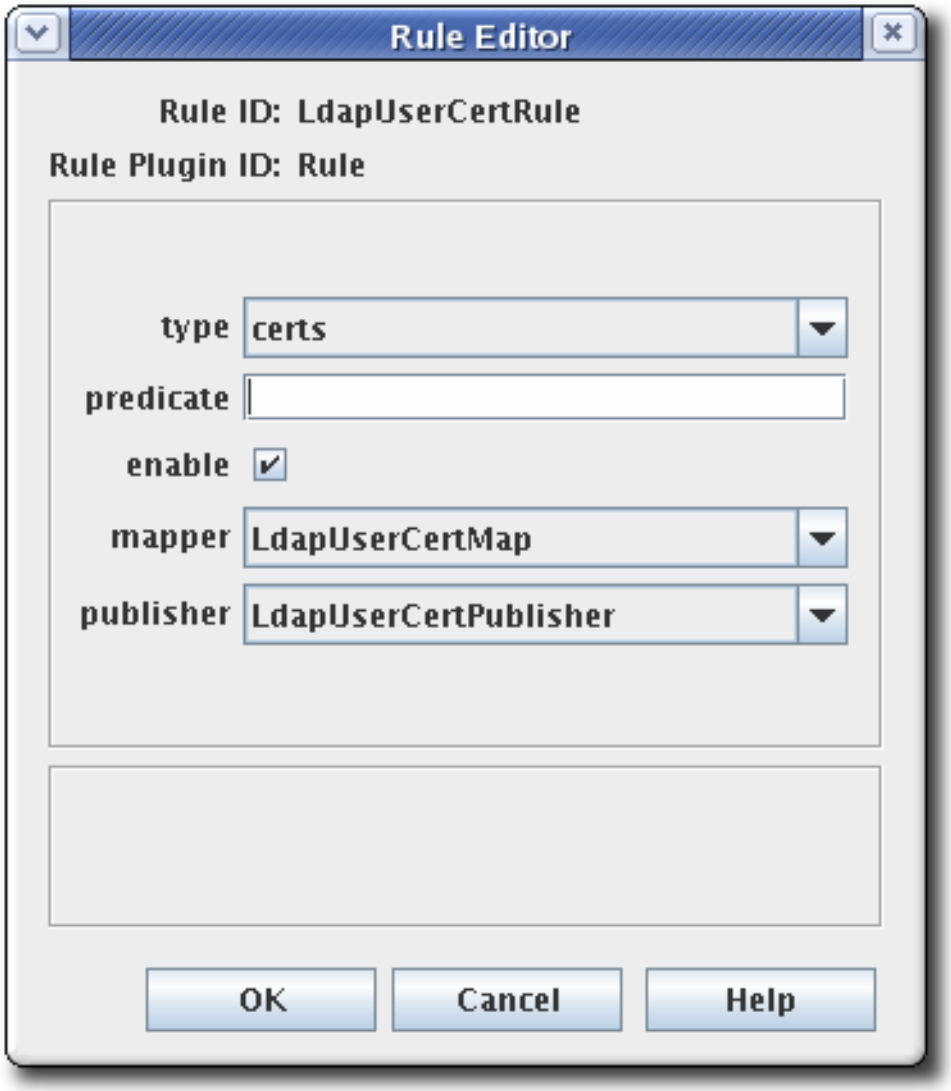
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布，然后选择 Rules。

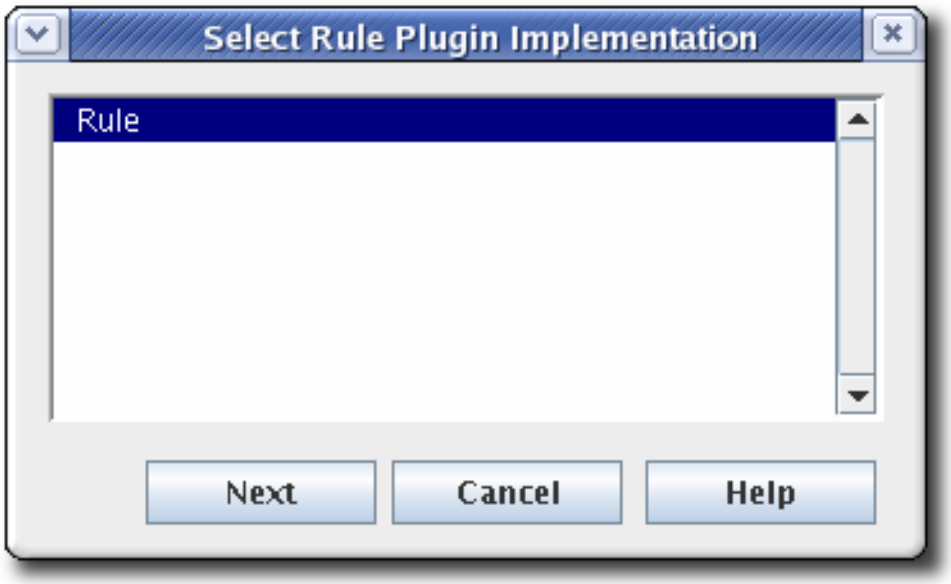
Rules Management 标签页（列出配置的规则）在右侧打开。



3. ***要编辑现有规则，请从列表中选择该规则，然后点 Edit。这将打开 Rule Editor 窗口。***

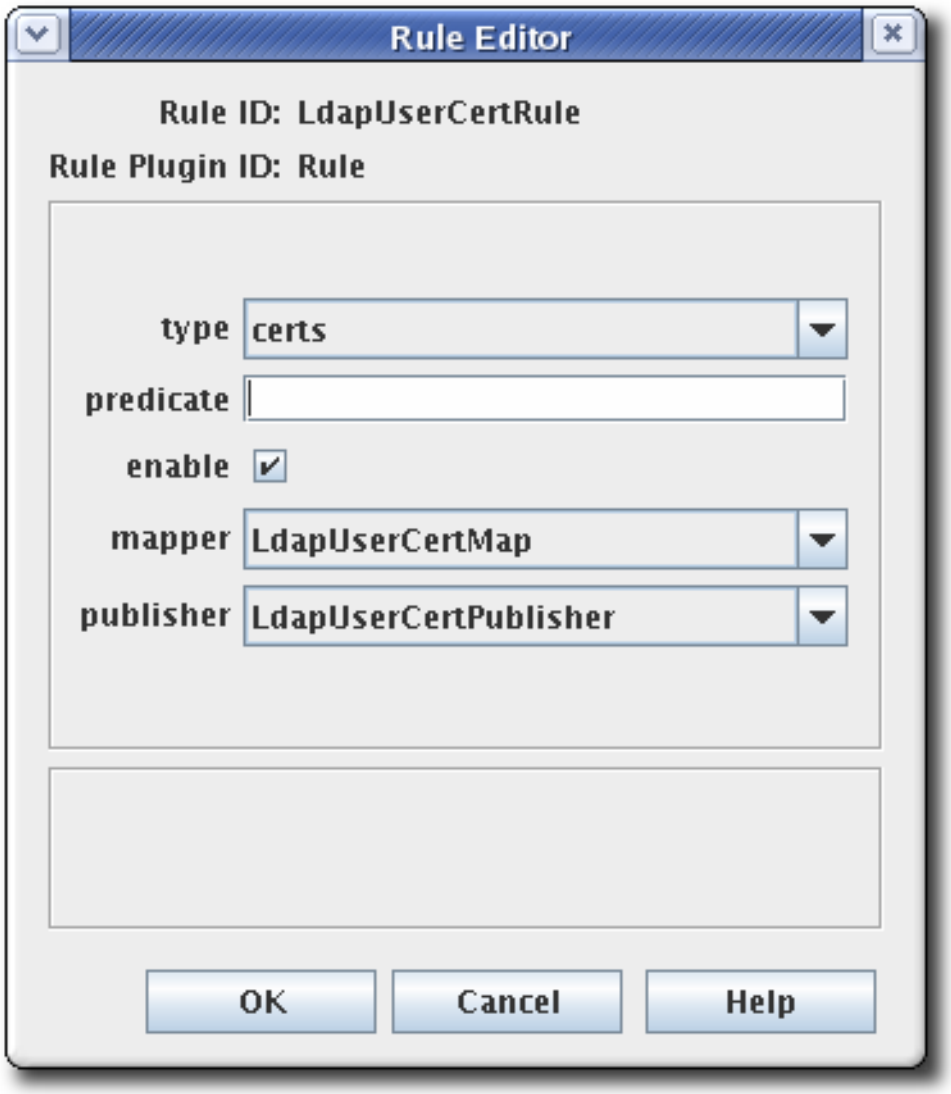


4. ***要创建规则，请单击 Add。这将打开 Select Rule Plug-in Implementation 窗口。***



*选择 **Rule** 模块。这是唯一默认模块。如果已经注册了任何自定义模块，它们也会可用。*

5. *编辑规则。*



- *键入。这是规则适用的证书类型。对于 CA 签名证书，值为 cacert。对于跨签名证书，值为 xcert。对于所有其他证书类型，值为 certs。对于 CRLs，指定 crl。*

- *predicate.这为规则应用到的证书或 CRL 发出类型设置 predicate 值。CRL 发出点、delta CRL 和证书的 predicate 值列在 表 8.3 “predicate 表达式” 中。*

- *启用。*

- *映射器.发布到文件时不需要映射程序，只有 LDAP 发布版需要它们。如果此规则与发布到 LDAP 目录的发布者关联，请在这里选择一个适当的映射程序。所有其他形式的发布留空。*

- *发布程序.将发布程序设置为与该规则关联。*

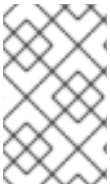
表 8.3 “predicate 表达式” 列出可用于识别 CRL 发出点和 delta CRLs 和证书配置集的 predicates。

表 8.3. predicate 表达式

predicate 类型	predicate
CRL 签发点	**issuingPointId==***Issuing_Point_Instance_ID* **&& isDeltaCRL==[true|false]** 要只发布 master CRL，设置为**DeltaCRL==false**。要只发布 delta CRL，请设置 **DeltaCRL==true**。要发布这两个情况，请为 master CRL 以及 delta CRL 中的一个规则设置一个规则。
证书配置集	**profileId==***profile_name* 要根据用来发布它们的配置集发布证书，请将 **profileId==** 设置为配置集名称，如 **caServerCert**。

8.6. 启用发布

只能针对文件（仅限 LDAP 或两者）启用发布。在设置发布者、规则和映射程序后，应启用发布程序。启用后，服务器会尝试开始发布。如果在启用前，如果发布没有被正确配置，发布可能会带来不所需行为，否则可能会失败。



注意

配置 CRL。必须配置 CRL，然后才能发布。请参阅 第 7 章 撤销证书和发布 CRL。

1. 登录到证书管理器控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布。

右窗格显示了发布到与 LDAP 兼容的目录的详细信息。

3. 要仅启用发布到文件，请选择 Enable Publishing。

4.

要启用 LDAP 发布，请选择 Enable Publishing and Enable Default LDAP Connection。

在 Destination 部分中，设置 Directory Server 实例的信息。

- *主机名.如果为 SSL 客户端通过身份验证的通信配置了 Directory 服务器，该名称必须与目录服务器的 SSL 服务器证书主体 DN 中的 cn 组件匹配。*

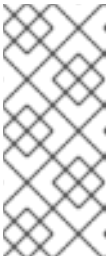
  主机名可以是完全限定域名或 IPv4 或 IPv6 地址。

- *端口号.*

- *目录管理器 DN.这是具有 Directory Manager 特权的目录条目的可分辨名称(DN)。证书管理器使用此 DN 访问目录树，并发布到 目录。为此 DN 设置的访问控制决定了证书管理器是否可以执行发布。可以创建另一个具有有限读写权限的 DN，只针对发布系统实际需要写入的属性。*

- *密码。这是 CA 用来绑定到发布证书或 CRL 的 LDAP 目录的密码。证书管理器将此密码保存在其 password.conf 文件中。例如：*

  ```
  CA LDAP Publishing:password
  ```



注意

标识发布密码(CA LDAP Publishing)的参数名称在 ca.publish.ldappublish.ldap.ldapauth.bindPWPrompt 参数中的 Certificate Manager 的 CS.cfg 文件中设置，并可编辑。

- 客户端证书。这将设置证书管理器用于在发布目录中进行 SSL 客户端身份验证的证书。默认情况下，证书管理器使用其 SSL 服务器证书。

- LDAP 版本.选择 LDAP 版本 3.

- 身份验证.证书管理器向 Directory 服务器进行身份验证的方式。选择是 基本身份验证 和 SSL 客户端身份验证。

  如果目录服务器是为基本身份验证或没有客户端身份验证的 SSL 通信而配置的，请选择 基本身份验证，并指定 Directory 管理器 DN 和密码的值。

  如果目录服务器被配置为与客户端身份验证进行 SSL 通信，请选择 SSL 客户端身份验证 和使用 SSL 通信 选项，并识别证书管理器必须用于 SSL 客户端身份验证的证书，以便对该目录进行 SSL 客户端身份验证。

服务器尝试连接到 Directory 服务器。如果信息不正确，服务器会显示错误消息。

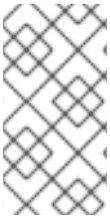
8.7. 启用发布队列

注册过程的一部分，包括向任何目录或文件发布发布证书。这基本上会关闭初始证书请求。但是，将证书发布到外部网络可能会显著降低了运行过程，这样会使请求处于打开状态。

为避免这种情况，管理员可以启用 发布队列。发布队列将发布操作（可能涉及外部 LDAP 目录）与请求和注册操作（使用单独的请求队列）分开。请求队列会立即更新，以显示注册过程已完成，而发布队列则以网络流量的速度发送信息。

发布队列设置已定义、有限的线程数发布生成的证书，而不是为每个批准的证书打开一个新线程。

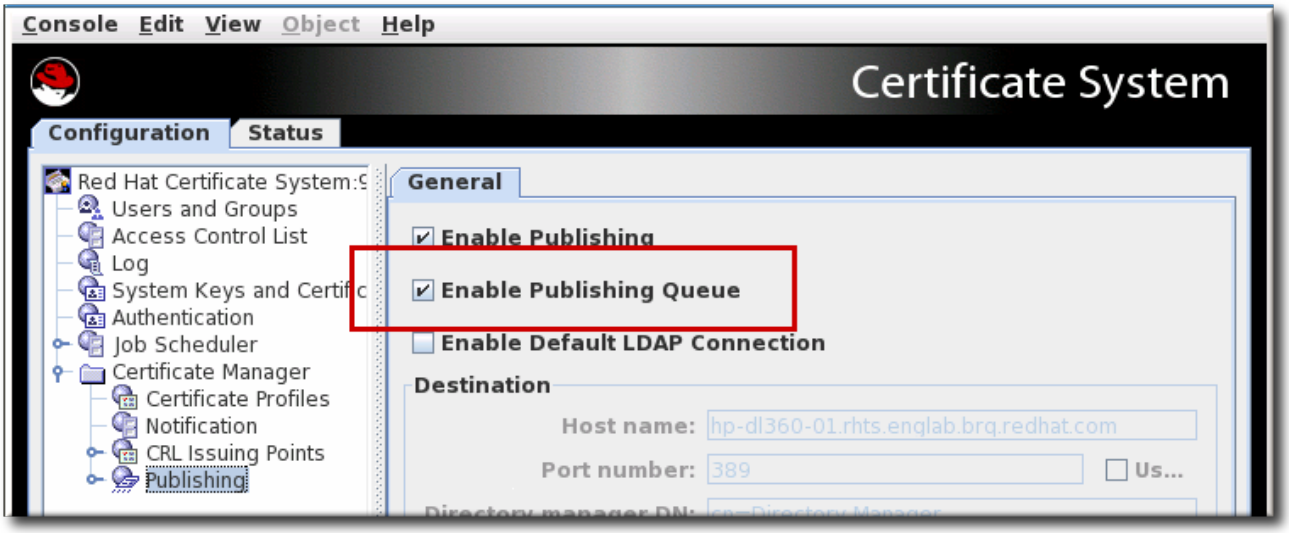
发布队列默认是禁用的。它可以在 CA 控制台中启用，同时启用发布。



注意

虽然发布队列默认为禁用，但 在控制台中 启用 LDAP 发布时，队列会自动启用。否则，可以手动启用队列。

图 8.1. 启用发布队列



提示

通过编辑 CS.cfg 文件启用发布队列可让管理员设置发布的选项，如用于发布操作的线程数量和队列页面大小。

有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅 红帽证书系统规划、安装和部署指南中的 启用 Publishing Queue 部分。

8.8. 设置恢复 CRL 下载

CertificateCertificate Systemnbsp;System 提供一些不中断的 CRL 下载方案。这可以通过通过 HTTP 发布 CRL 作为普通文件来完成。这种下载 CRL 的方法提供了检索 CRL 和降低整体网络拥塞的灵活性。

8.8.1. 使用 wget 检索 CRL

由于 CRL 可通过 HTTP 作为文本文件发布，因此可以使用 wget 等工具从 CA 手动检索。wget 命令可用于检索任何已发布的 CRL。例如，要检索一个比前面的完整 CRL 的最新 CRL：

```
[root@server ~]# wget --no-check-certificate -d
https://server.example.com:8443/ca/ee/ca/crl/MasterCRL.bin
```

表 8.4 “wget 选项用于检索 CRL” 中总结了 wget 的相关参数。

表 8.4. wget 选项用于检索 CRL

参数	描述
no 参数	检索完整的 CRL。
-N	检索比本地副本(delta CRL)更新的 CRL。
-c	检索部分下载的文件。
--no-check-certificate	跳过连接的 SSL，因此不需要在主机和客户端之间配置 SSL。
-d	打印调试信息。

8.9. 发布跨PAIR 证书

跨对证书可以作为 跨CertificatePair 条目发布到 LDAP 目录或文件，这默认是启用的。如果禁用了它，可以通过证书管理器控制台重新启用它：

1. *打开 CA 控制台。*

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. *在 Configuration 选项卡中，选择左侧窗格中的 证书管理器 链接，然后选择 Publishing 链接。*

3. *单击 发布 下的 Rules 链接。这会打开右侧的 Rules Management 窗格。*

4. *如果规则存在并且已禁用，请选择 启用 复选框。如果规则已被删除，请单击 Add 并创建一个新规则。*

a. 从类型下拉菜单中选择 xcerts。

b. 确保选中了 启用 复选框。

c. 从 mapper 下拉菜单中选择 LdapCaCertMap。

d. 从 发布程序 下拉菜单中选择 LdapCrossCertPairPublisher。

发布规则中指定的 映射程序 和 发布程序 均列在 CA 控制台左侧导航窗口的 Publishing 下方。默认情况下，映射程序 LdapCaCertMap 指定将 crossCertificatePair 存储到 LdapCaSimpleMap LDAP 条目。默认情况下，发布程序(LDAPCrossPairPublisher)设置属性，以在 CA 条目中将跨密钥对证书存储在 CA 条目中，以 跨CertificatePair;binary。

有关使用跨对证书的更多信息，请参阅 第 16.5 节 “使用跨Pair 证书”。

有关创建跨对证书配置集的更多信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 配置跨Pair 配置集 部分。

8.10. 测试发布到文件

验证证书管理器是否在正确发布证书和 CRL 以进行文件：

1. 打开 CA 的最终用户页面，再请求证书。
2. 如果需要，请通过代理服务页面批准请求。
3. 从终端页面检索证书，并将证书下载到浏览器中。
4. 检查服务器是否生成了包含证书的 DER 编码文件。

打开应发布证书的二进制 blob 的目录。证书文件应命名为 cert-serial_number.der。

5. 使用 Binary Binary 将 DER 编码的证书转换为其基本 64 编码的格式。有关此工具的更多信息，请参阅 BtoA(1) man page。

```
BtoA input_file output_file
```

input_file 设置包含 DER 编码证书的文件的路径，而 output_file 设置文件的路径，以写入 base-64 编码证书。

6. 打开 ASCII 文件; base-64 编码证书与所示的证书类似：

```
-----BEGIN CERTIFICATE-----
MMIIBtgYJYIZIAYb4QgIFoIIBpzCCAZ8wggGbMIIBRaADAgEAAgEBMA0GCSqGSIb3DQEBB
AUAMFcxC
AJBgNVBAYTAIVTMSwwKgYDVQQKEyNOZXRzY2FwZSBDb21tdW5pY2F0aWhfyyuougjgjjg
mkgjkgmjg
fjfgjjjgfyjfyj9ucyBDb3Jwb3JhdGlvbjpMEaMBgGA1UECxMRSXNzdWluZyhgdfhbfdpffjphotoo
gdhkBBdXRob3JpdHkwHhcNOTYxMTA4MDkwNzM0WhcNOTgxMTA4MDkwNzMM0WjBXMQ
swCQYDVQQGEwJ
VUzEsMCoGA1UEChMjTmV0c2NhcGUgQ29tbXVuaWNhdGlvbnMgQ29ycG9yY2F0aW9ucyB
Db3Jwb3Jhd
GlvbjpMEaMBgGA1UECxMRSXNzdWluZyBBdXRob3JpdHkwHh
-----END CERTIFICATE-----
```

7. 使用 Pretty Print Certificate 工具将基本 64 编码的证书转换为可读的表单。有关此工具的更多信息，请参阅 PrettyPrintCert(1) man page。

```
PrettyPrintCert input_file [output_file]
```

input_file 设置 ASCII 文件的路径，其中包含 base-64 编码证书，以及 output_file （可选）设置要写入证书的路径。如果没有设置输出文件，证书信息将写入到标准输出中。

8. 将输出与发布的证书进行比较；检查证书中的序列号和文件名中使用的序列号。

如果一切匹配，证书管理器将正确配置为将文件发布证书。

9. 撤销证书。

10. 检查服务器是否生成了包含 CRL 的 DER 编码文件。

    打开服务器将 CRL 发布为二进制 blob 的目录。CRL 文件应当具有格式为 crl-this_update.der 的名称。this_update 指定从 CRL 依赖时间的 This Update 变量获取的值。

11. 使用 Binary Binary 将 DER 编码的 CRL 转换为其基本 64 编码的格式。

```
BtoA input_file output_file
```

12. 使用 Pretty Print CRL 工具，将 base 64 编码的 CRL 转换为可读格式。

```
PrettyPrintCrl input_file [output_file]
```

13. 比较输出。

8.11. 查看证书和 CRLS 发布到文件

证书和 CRL 可以发布到两类文件：base-64 编码或 DER-encoded。可以使用 dumpasn1 工具或 PrettyPrintCert 或 PrettyPrintCrl 工具，查看这些文件的内容。

查看 base-64 编码文件中的内容：

1. 将 base-64 文件转换为二进制文件。例如：

```
AtoB /tmp/example.b64 /tmp/example.bin
```

2. 使用 PrettyPrintCert 或 PrettyPrintCrl 工具将二进制文件转换为 prettyprint 格式。例如：

```
PrettyPrintCert example.bin example.cert
```

要查看 DER 编码文件的内容，只需运行 dumpasn1、 PrettyPrintCert 或 PrettyPrintCrl 工具（使用 DER 编码文件）。例如：

```
PrettyPrintCrl example.der example.crl
```

8.12. 更新目录中的证书和 CRL

如果在 Directory 服务器停机期间签发或撤销证书，则证书管理器和发布目录可能会不同步。发布或撤销的证书需要在 Directory 服务器备份时手动发布或取消发布。

要找到与目录不同步的证书 - 不在目录中有效证书，并撤销了仍在目录中或过期的证书 - 证书管理器会在其内部数据库中保留一条记录，无论其内部数据库中的证书是否已发布到该目录。如果证书管理器和发布目录未同步，请使用证书管理器代理服务页面中的 Update Directory 选项，将发布目录与内部数据库同步。

以下选择可用于与内部数据库同步目录：

- 搜索不同步和发布或未发布或取消发布的证书的内部数据库。
- 发布在 Directory 服务器停机期间发布的证书。同样，未发布在 Directory 服务器停机期间被撤销或过期的证书。
- 根据序列号发布或取消发布范围证书，从序列号 xx 向序列号 yy。

证书管理器的发布目录只能由证书管理器代理手动更新。

8.12.1. 手动更新目录中的证书

证书管理器代理服务页面中的 Update Directory Server 表单可以用来使用与证书相关的信息手动更新目录。这个表单启动以下操作的组合：

- 使用证书更新目录。
- 从目录中删除过期的证书。

通过调度自动作业，可以从发布目录中删除过期的证书。详情请查看 第 12 章 设置自动化任务。

- 从目录中删除撤销的证书。

通过执行以下操作手动更新目录：

1. 打开证书管理器代理服务页面。
2. 选择 Update Directory Server 链接。
3. 选择适当的选项，然后单击 Update Directory。

证书管理器开始使用其内部数据库中的证书信息来更新目录。如果更改非常大，更新目录可能需要较长时间。在此期间，通过证书管理器进行的任何更改，包括发布任何证书或被撤销的任何证书，则可能不会包含在更新中。如果在更新目录的同时发布或撤销任何证书，请再次更新目录以反映这些更改。

目录更新完成后，证书管理器会显示状态报告。如果进程中断，服务器会记录错误消息。

如果证书管理器作为 root CA 安装，则在使用代理接口更新带有有效证书的目录时，可以使用发布规则为用户证书发布 CA 签名证书。这可能会返回一个对象类违反错误，或者映射器中的其他错误。选择适当的序列号范围来排除 CA 签名证书，可以避免此问题。CA 签名证书是第一个证书 root CA 问题。

- 通过将 predicate 参数的值更改为 profileId!=caCACert 来修改用户证书的默认发布规则。
- 使用 LdapCaCertPublisher publisher 插件模块添加另一个规则，使用 predicate 参数设置为 profileId=caCACert，以发布子协调 CA 证书。

8.12.2. 手动更新目录中的 CRL

证书管理器 代理服务页面中的证书撤销列表表单需要使用 CRL 相关信息手动更新目录。

通过执行以下操作手动更新 CRL 信息：

1. 打开证书管理器代理服务页面。

2. 选择 Update Revocation List。

3. 点 Update。

证书管理器开始在其内部数据库中使用 CRL 更新目录。如果 CRL 较大，更新目录需要相当长的时间。在此期间，对 CRL 所做的任何更改可能不会包含在更新中。

更新目录时，证书管理器会显示状态报告。如果进程中断，服务器会记录错误消息。

8.13. 注册自定义映射程序和发布程序插件模块

可以在证书管理器的发布框架中注册新的映射程序或发布程序插件模块。可以删除不需要的 mapper 或 publisher 插件模块。在删除模块之前，请删除基于此模块的所有规则。

1. 创建自定义作业类。在本例中，自定义发布程序插件名为 MyPublisher.java。

2. 编译新类。

```
javac -d . -classpath $CLASSPATH MyPublisher.java
```

3. 在 CA 的 WEB-INF Web 目录中创建用于存放自定义类的目录，以便 CA 可以访问它们。

```
mkdir /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

4. 将新插件文件复制到 新类 目录中，并将所有者设置为证书证书系统nbsp;System 系统用户 (pkiuser)。

```
cp -pr com /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes

chown -R pkiuser:pkiuser /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

5. **注册插件。**

   a. **登录到证书管理器控制台。**

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

   b. **在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 发布。**

   c. **要注册映射程序模块，请选择 映射程序，然后选择 映射程序插件注册 选项卡。**

   要注册发布程序模块，请选择 publishers，然后选择 publisher Plug-in Registration 选项卡。

   d. **要注册插件，请点击 Register。**

   e. **设置插件名称和插件类名称。类名称是实施 Java 类的路径。如果这个类是软件包的一部分，请包含软件包名称。例如，要在名为 com.customplugins 的软件包中注册名为 customMapper 的类，其名称是 com.customplugins.customMapper。**

第 9 章 注册证书的身份验证

本章论述了如何注册端到端实体证书、如何创建和管理服务器证书、证书证书系统中的身份验证方法；系统在注册端点实体证书时使用，以及如何设置这些身份验证方法。

注册 即向最终实体发布证书的过程。该进程正在创建和提交请求，对请求进行身份验证，然后批准请求并发布证书。

用于验证最终实体的方法决定了整个注册过程。证书Certificate Systemnbsp;System 可以验证实体的方法有三种：

- 在 代理批准的 注册中，终端请求将发送到代理进行批准。代理批准证书请求。
- 在自动注册中，终端请求通过插件进行身份验证，然后处理证书请求；不会参与注册流程中的代理。
- 在 CMC 注册 中，第三方应用程序可以创建由代理签名并自动处理的请求。

最初为代理批准的注册和 CMC 身份验证配置证书管理器。通过配置其中一个身份验证插件模块来启用自动注册。可以在子系统的单一实例中配置多个身份验证方法。

注意

通过配置自动通知，可为任何身份验证方法签发证书时，可将电子邮件自动发送到终端实体。有关通知的更多信息，请参阅 第 11 章 使用自动通知。

9.1. 配置代理(APPROVED ENROLLMENT)

证书管理器最初是为代理批准的注册配置。最终实体发出请求，请求发送到代理队列以进行代理批准。代理可以修改请求，更改请求的状态，拒绝请求或批准请求。批准请求后，签署的请求将发送到证书管理器进行处理。证书管理器处理请求并发布证书。

代理批准的注册方法不可配置。如果没有为任何其他注册方法配置证书管理器，服务器会自动将所有与证书相关的请求发送到其等待代理批准的队列。这样可确保所有缺少身份验证凭据的请求都发送到请求队列进行代理批准。

要使用代理批准的注册，请将验证方法 留空。例如：

```
auth.instance_id=
```

9.2. 自动注册

在自动注册中，用户已成功通过身份验证插件模块中设定的方法验证验证后，会立即处理终端注册请求；不需要代理批准。提供了以下身份验证插件模块：

- 基于目录的注册。最终实体使用其用户 ID 和密码其 DN 和密码通过 LDAP 目录进行身份验证。请参阅 第 9.2.1 节 “设置基于目录的身份验证”。
- 基于 PIN 的注册。使用其目录 ID、密码和 PIN 在目录条目中设置的 PIN，对最终用户实体进行身份验证。请参阅 第 9.2.2 节 “设置基于 PIN 的注册”。
- 基于证书的身份验证.某种实体（如服务器或令牌）的实体使用 CA 发布的证书向 CA 进行身份验证，该实体证明其身份。这最常用于续订，其中显示原始证书以验证续订过程。请参阅 第 9.2.3 节 “使用基于证书的验证”。
- AgentCertAuth.如果提交请求的实体作为子系统代理进行身份验证，此方法会自动批准证书请求。用户通过提供代理证书作为代理进行身份验证。如果所提供的证书由子系统识别为代理证书，则 CA 会自动处理证书请求。

  此形式的自动身份验证可与用于注册服务器证书的证书配置集关联。

  此插件默认启用，且没有参数。
- 基于文件格式的注册.用于路由器(SCEP)注册，会使用一个文本文件，其中包含 IP 地址、主机名或其他标识符列表，它是一个随机的 PIN。路由器使用其 ID 和 PIN 验证 CA，然后 CA 会将提供的凭证与文本文件中的身份列表进行比较。请参阅 第 9.2.4 节 “配置平面文件身份验证”。

9.2.1. 设置基于目录的身份验证

UidPwdDirAuth 和 UdnPwdDirAuth 插件模块实现基于目录的身份验证。最终用户通过在 LDAP 目录中提供用户 ID 或 DN 和密码来注册证书。

1. 创建 UidPwdDirAuth 或 UdnPwdDirAuth 身份验证插件模块并配置实例的实例。

   a. 打开 CA 控制台。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

   b. 在 Configuration 选项卡中，在导航树中选择 Authentication。

   右窗格显示 Authentication Instance 选项卡，它列出了当前配置的验证实例。

   注意

   UidPwdDirAuth 插件默认启用。

   c. 点 Add。

   此时会出现 Select Authentication Plug-in Implementation 窗口。

   d. 为用户 ID 和密码身份验证选择 UidPwdDirAuth，或者选择 UdnPwdDirAuth 用于 DN 和密码身份验证。

   e. 在 Authentication Instance Editor 窗口中填写以下字段：

   - 身份验证实例 ID。接受默认实例名称，或输入新名称。
   - dnpattern.指定一个字符串，代表主题名称模式，以便根据目录属性和条目 DN 进行公式限制。
   - ldapStringAttributes.指定端点实体应被视为身份验证的 LDAP 字符串属性列表。如果指定，与这些属性对应的值将从身份验证目录中复制到身份验证令牌，并由证书

配置集用于生成主题名称。此参数输入值是可选的。

- IdapByteAttributes.指定端点实体应被视为身份验证的 LDAP 字节（二进制）属性列表。如果指定，与这些属性对应的值将从身份验证目录中复制到身份验证令牌，供其他模块使用，如向用户证书添加额外信息。

  此参数输入值是可选的。

- ldap.ldapconn.host.指定身份验证目录的完全限定 DNS 主机名。

- ldap.ldapconn.port.指定验证目录侦听请求的 TCP/IP 端口；如果选择了 ldap.ldapconn.secureConn. 复选框，则应为 SSL 端口号。

- ldap.ldapconn.secureConn.指定身份验证目录监听来自证书证书系统nbsp 的请求的端口类型 SSL 或非 SSL；系统.如果这是 SSL 端口，请选择此项。

- ldap.ldapconn.version.指定 LDAP 协议版本，可以是 2 或 3。默认值为 3，因为版本 3.x 以后所有目录服务器都是 LDAPv3。

- ldap.basedn.指定用于搜索身份验证目录的基本 DN。服务器使用 HTTP 输入的 uid 字段的值（用户在注册表单中输入的用户）和基本 DN 来构造 LDAP 搜索过滤器。

- ldap.minConns.指定身份验证目录允许的最小连接数。允许的可能值为 1 到 3。

- ldap.maxConns.指定身份验证目录允许的最大连接数。可见的值有 3 到 10。

f. 点击 OK。身份验证实例已设置并启用。

2. 通过为特定证书设置策略，设置证书配置集用于注册用户。通过在证书配置集中配置输入来自定义注册表单，并包括插件验证用户身份所需的信息输入。如果默认输入不包含需要收集的所有信息，请提交使用第三方工具创建的请求。

   有关配置配置集的详情请参考 第 3.7.2 节 “在 Subject Alt Name 中插入 LDAP 目录属性值

和其他信息”。

设置绑定 LDAP 连接

有些环境需要禁止匿名绑定用于身份验证的 LDAP 服务器。要在 CA 和 LDAP 服务器之间建立绑定连接，您需要进行以下配置更改：

- 根据 CS.cfg 中的以下示例设置基于目录的身份验证：

```
auths.instance.UserDirEnrollment.ldap.ldapBoundConn=true
auths.instance.UserDirEnrollment.ldap.ldapauth.authtype=BasicAuth
auths.instance.UserDirEnrollment.ldap.ldapauth.bindDN=cn=Directory Manager
auths.instance.UserDirEnrollment.ldap.ldapauth.bindPWPrompt=externalLDAP
externalLDAP.authPrefix=auths.instance.UserDirEnrollment
cms.passwordlist=internaldb,replicationdb,externalLDAP
```

其中 bindPWPrompt 是 password.conf 文件中使用的标签或提示；它也是 optionpasswordlist 和 authPrefix 选项下使用的名称。

- 在 password.conf 中添加来自 CS.cfg 的标签或提示：

```
externalLDAP=your_password
```

设置外部授权

还可配置基于目录的身份验证插件来评估用户的组成员资格进行身份验证。要以这种方式设置插件，必须在 CS.cfg 中配置以下选项：

- groupsEnable 是一个布尔值选项，用于启用检索组。默认值为 false。
- 基于组 的基本 DN。当它不同于 基于默认n 时，需要指定它。
- 组是 组的 DN 组件。默认值为 ou=groups。
- groupObjectClass 是以下组对象类之一： groupofquenames、groupofnames。默认值为 groupofquenames。

- *groupUseridName 是 group object member 属性中用户 ID 属性的名称。默认值为 cn。*

- *useridName 是用户 ID DN 组件的名称。默认值为 uid。*

- *searchGroupUserByUserdn 是一个布尔值选项，决定是否搜索 userdn 或 ${groupUserIdName}=${uid} 属性的组对象 member 属性。默认值为 true。*

例如：

```
auths.instance.UserDirEnrollment.pluginName=UidPwdDirAuth
auths.instance.UserDirEnrollment.ldap.basedn=cn=users,cn=accounts,dc=local
auths.instance.UserDirEnrollment.ldap.groupObjectClass=groupofnames
auths.instance.UserDirEnrollment.ldap.groups=cn=groups
auths.instance.UserDirEnrollment.ldap.groupsBasedn=cn=accounts,dc=local
auths.instance.UserDirEnrollment.ldap.groupsEnable=true
auths.instance.UserDirEnrollment.ldap.ldapconn.host=local
auths.instance.UserDirEnrollment.ldap.ldapconn.port=636
auths.instance.UserDirEnrollment.ldap.ldapconn.secureConn=true
```

最后，您必须修改 /instance_path/ca/profiles/ca/profile_id.cfg 文件，以配置配置集以使用 CS.cfg 中定义的 UserDirEnrollment auth 实例，并在适当的情况下为基于组的授权提供 ACL。例如：

```
auth.instance_id=UserDirEnrollment
auths.acl=group="cn=devlab-access,ou=engineering,dc=example,dc=com"
```

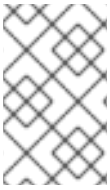
9.2.2. 设置基于 PIN 的注册

基于 PIN 的身份验证涉及为 LDAP 目录中的每个用户设置 PIN，将这些 PIN 分发到用户，然后让用户在填写证书请求时提供 PIN 和 password。然后，用户使用其用户 ID 和密码 LDAP 条目对 LDAP 目录进行身份验证，并使用其 LDAP 条目中的 PIN 进行验证。当用户成功验证时，请求会自动处理，并发出新证书。

CertificateCertificate Systemnbsp;System 提供了一个工具 setpin，它为 Directory Server 添加所需的 schema，并为每个用户生成 PIN。

PIN 工具执行以下功能：

- 在 LDAP 目录中为 PIN 添加所需的 schema。
- 在设置的 PIN manager 用户中添加具有读写权限的 PIN manager 用户。
- 设置 ACI 以允许在使用 PIN 后删除 PIN，为 PIN 提供读写权限，并阻止用户创建或更改 PIN。
- 在每个用户条目中创建 PIN。



注意

此工具记录在 证书系统命令行工具指南 中。

1. 使用 PIN 工具添加 PINs 所需的 schema，将 PIN 添加到用户条目，然后将 PINs 分发到用户。

   a. 打开 /usr/share/pki/native-tools/ 目录。

   b. 在文本编辑器中打开 setpin.conf 文件。

   c. 按照文件中概述的说明进行操作，并进行适当的更改。

      通常，需要更新的参数是 Directory Server 的主机名、Directory Manager 的绑定密码以及 PIN 管理器的密码。

   d. 使用 optfile 选项运行 setpin 命令，指向 setpin.conf 文件。

```
setpin optfile=/usr/share/pki/native-tools/setpin.conf
```

      该工具使用新属性（默认为 pinPerson）和一个新的对象类（默认为 pinPerson）修改 schema，并创建一个 pinmanager 用户，并只设置 ACI 以允许 pinmanager 用户修改 pin 属性。

e.

要为特定用户条目生成 PIN，或提供用户定义的 PIN，创建一个输入文件，其中包含列出的条目的 DN。对于 ezample：

```
dn:uid=bjensen,ou=people,dc=example,dc=com
dn:uid=jsmith,ou=people,dc=example,dc=com
dn:jtyler,ou=people,dc=example,dc=com
...
```

有关构建输入文件的详情，请参考 证书系统命令行工具指南中的 PIN 生成器章节。

f.

禁用 setpin 命令的设置模式。注释掉 setup 行，或者将值更改为 no。

```
vim /usr/share/pki/native-tools/setpin.conf

setup=no
```

设置模式会创建所需的 uers 和对象类，但工具在设置模式下不会生成 PIN。

g.

运行 setpin 命令在目录中创建 PIN。

提示

在不使用 write 选项的情况下运行该工具来生成 PIN 列表，而无需实际更改该目录。

例如：

```
setpin host=yourhost port=9446 length=11 input=infile output=outfile write
"binddn=cn=pinmanager,o=example.com" bindpw="password" basedn=o=example.com
"filter=(uid=u*)" hash=sha256
```

WARNING

不要将 hash 参数设置为 none。使用 hash=none 运行 setpin 命令会导致 pin 以纯文本形式存储在用户 LDAP 条目中。

h. 完成设置所需的身份验证方法后，使用输出文件向用户提供 PIN。

确认基于 PIN 的注册工作后，将 PIN 提供给用户，以便在注册期间使用它们。要保护 PIN 的隐私，请使用安全的带外交付方法。

2. 在证书配置集中设置特定证书的策略以注册用户。有关证书策略的详情，请参阅 第 3 章 制作发行证书规则（证书配置文件）。

3. 创建并配置 UidPwdPinDirAuth 身份验证插件的实例。

a. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

b. 在 Configuration 选项卡中，在导航树中选择 Authentication。

右窗格显示 Authentication Instance 选项卡，它列出了当前配置的验证实例。

c. 点 Add。

此时会出现 Select Authentication Plug-in Implementation 窗口。

d. 选择 UidPwdPinDirAuth 插件模块。

e.

在 Authentication Instance Editor 窗口中填写以下字段：

- 身份验证实例 ID。接受默认实例名称或输入新名称。
- removePin.设置在最终用户成功验证后是否从身份验证目录中删除 PIN。从目录中删除 PIN 会限制用户多次注册，从而防止他们获得多个证书。
- pinAttr.指定 PIN 的身份验证目录属性。PIN Generator 程序将 属性设置为 setpin.conf 文件中的 objectclass 参数的值；此参数的默认值是 pin。
- dnpattern.指定一个字符串，代表主题名称模式，以便根据目录属性和条目 DN 进行公式限制。
- ldapStringAttributes.指定端点实体应被视为身份验证的 LDAP 字符串属性列表。此参数输入值是可选的。
- ldapByteAttributes.指定端点实体应被视为身份验证的 LDAP 字节（二进制）属性列表。如果指定，与这些属性对应的值将从身份验证目录中复制到身份验证令牌，供其他模块使用，如向用户证书添加额外信息。

  此参数输入值是可选的。
- ldap.ldapconn.host.指定身份验证目录的完全限定 DNS 主机名。
- ldap.ldapconn.port.指定身份验证目录侦听来自证书证书系统nbsp 的 TCP/IP 端口；系统.
- ldap.ldapconn.secureConn.指定身份验证目录侦听请求的端口的类型 SSL 或非 SSL。如果这是 SSL 端口，请选择此项。
- ldap.ldapconn.version.指定 LDAP 协议版本，可以是 2 或 3。默认情况下，这是 3，因为除 3.x 之后的所有目录服务器版本都是 LDAPv3。

- *ldap.ldapAuthentication.bindDN.指定在从身份验证目录中删除 PIN 时要绑定的用户条目。仅在选择了 removePin 复选框时指定此参数。建议有一个单独的用户条目，该用户条目只能修改目录中创建和使用 目录中的 PIN 属性。例如，不要使用 Directory Manager 的条目，因为它具有修改整个目录内容的特权。*

- *密码。指定与 ldap.ldapauthbindDN 参数指定的 DN 关联的密码。保存更改后，服务器会将密码存储在单点登录密码缓存中，并使用它来后续启动。只有在选择了 removePin 复选框时，才需要设置此参数。*

- *ldap.ldapAuthentication.clientCertNickname.指定用于对认证目录进行 SSL 客户端身份验证的证书 nickname 来删除 PIN。确保证书有效且已由身份验证目录的证书数据库中信任的 CA 签名，并且身份验证目录的 certmap.conf 文件已配置为将证书正确映射到目录中的 DN。只适用于 PIN 才可以删除。*

- *ldap.ldapAuthentication.authtype.指定从身份验证目录中删除 PIN 所需的身份验证类型、基本身份验证或 SSL 客户端身份验证。*

  - *Basic auth 指定基本身份验证。使用此选项时，为 ldap.ldapAuthentication.bindDN 和 password 参数输入正确的值，服务器使用 ldap.ldapAuthentication.bindDN 属性中的 DN 绑定到该目录。*

  - *SslClientAuth 指定 SSL 客户端身份验证。使用此选项时，将 ldap.ldapconn.secureConn 参数的值设置为 true，将 ldap.ldapAuthentication.clientCertNickname 参数的值设置为用于 SSL 客户端身份验证的 nickname。*

- *ldap.basedn.指定用于搜索身份验证目录的基本 DN；服务器使用 HTTP 输入中的 uid 字段的值（用户在注册表单中输入的用户）和基本 DN 来构造 LDAP 搜索过滤器。*

- *ldap.minConns.指定身份验证目录允许的最小连接数。允许的可能值为 1 到 3。*

- *ldap.maxConns.指定身份验证目录允许的最大连接数。可见的值有 3 到 10。*

f. *点击 OK。*

4.

通过在证书配置集中配置输入来自定义注册表单。包括插件验证用户身份所需的信息。如果默认输入不包含需要收集的所有信息，请提交使用第三方工具创建的请求。

9.2.3. 使用基于证书的验证

提供的 证书认证是验证请求者的身份并自动验证提交的请求时，基于证书的身份验证。这通常用于续订过程，当用户、服务器和应用程序提供原始证书时，会使用该证书来验证请求。

在最初请求证书时，使用基于证书的身份验证可能很有用。例如，令牌可以批量加载通用证书，然后用于对用户注册用户证书时验证用户，或者，也可以向用户签发证书，以供用户用于验证其加密证书的请求。

基于证书的验证模块(SSLclientCertAuth)默认是启用的，这个验证方法可在任何自定义证书配置集中引用。

9.2.4. 配置平面文件身份验证

使用随机生成的 PIN 注册并进行身份验证。CA 使用 flatFileAuth 身份验证模块处理包含路由器身份验证凭据的文本文件。

9.2.4.1. 配置 flatFileAuth 模块

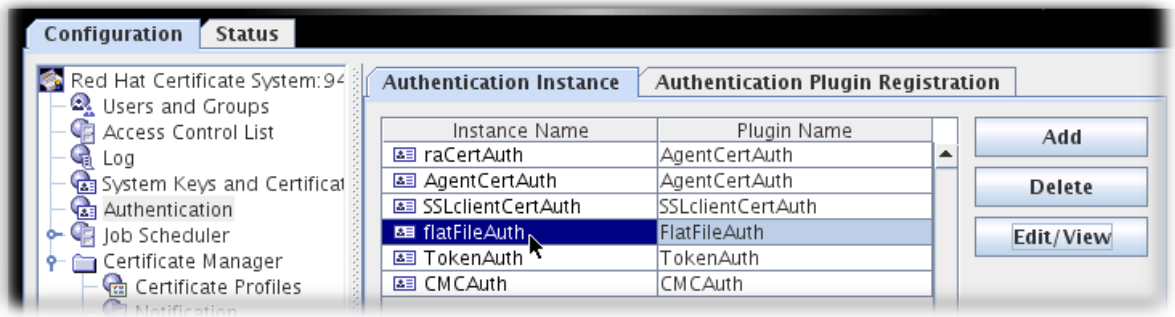
已为 SCEP 注册配置了无格式文件身份验证，但可以编辑平面文件及其身份验证参数的位置。

1. 打开 CA 控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，在导航树中选择 Authentication。

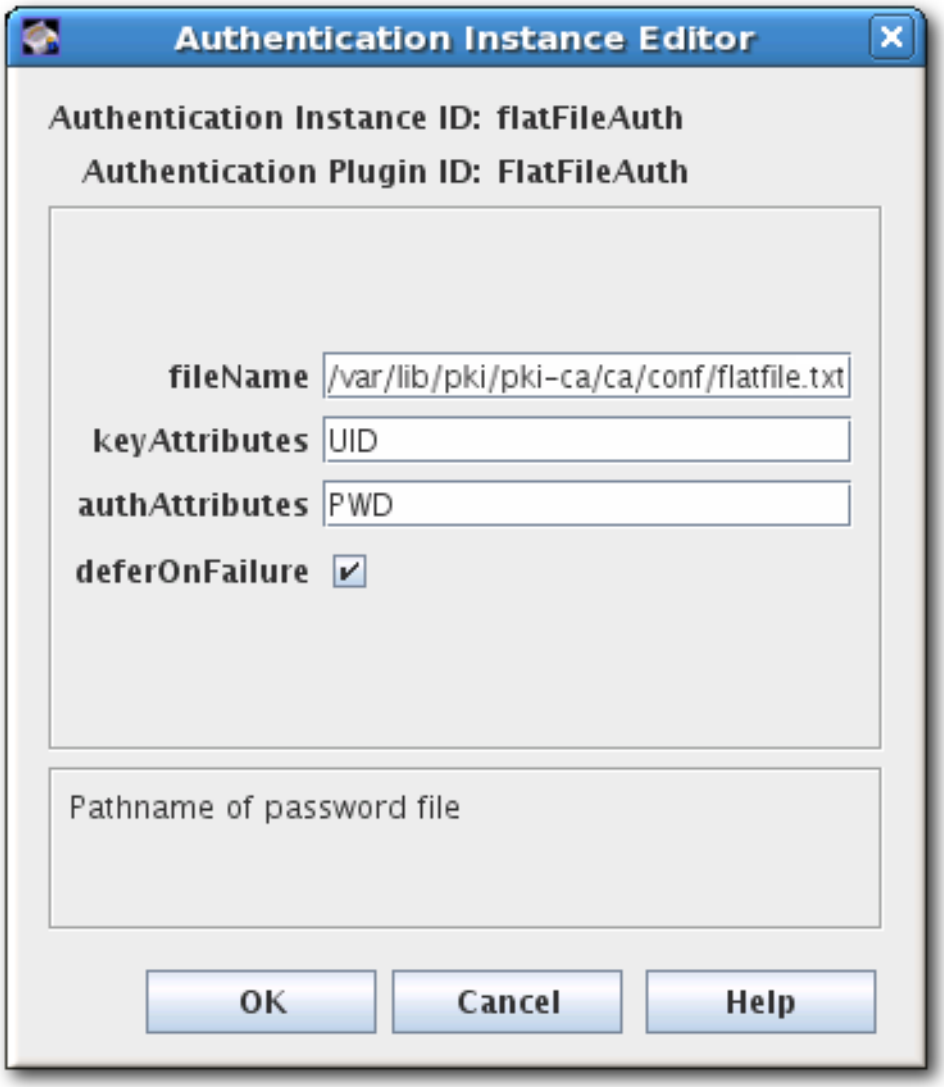
3. 选择 flatFileAuth 身份验证模块。



4. *单击 编辑/查看。*

5. ***要更改文件位置和名称，请重置 fileName 字段。***

要更改身份验证名称参数，请将 keyAttributes 值重置为 SCEP 注册表单中提交的另一个值，如 CN。也可以通过逗号（如 UID、CN ）来分隔多个名称参数，也可以使用多个名称参数。要更改 password 参数名称，请重置 authAttributes 字段。



6. 保存编辑。

9.2.4.2. 编辑 flatfile.txt

相同的 flatfile.txt 文件用于验证每个 SCEP 注册。每次向路由器发布新的 PIN 时，都必须手动更新此文件。

默认情况下，这个文件位于 /var/lib/pki/pki-ca/ca/ 中，并为每个身份验证条目指定两个参数，站点的 UID（通常是它的 IP 地址，可以是 IPv4 或 IPv6），以及路由器发出的 PIN。

```
UID:192.168.123.123
PIN:HU89dj
```

每个条目都必须跟一个空行。例如：

```
UID:192.168.123.123
PIN:HU89dj

UID:12.255.80.13
PIN:fiowIO89

UID:0.100.0.100
PIN:GRIOjisf
```

如果身份验证条目没有用空行分隔，那么当路由器尝试对 CA 进行身份验证时，将失败。例如：

```
... flatfile.txt entry ...
UID:192.168.123.123
PIN:HU89dj
UID:12.255.80.13
PIN:fiowIO89

... error log entry ...
[13/Jun/2020:13:03:09][http-9180-Processor24]: FlatFileAuth: authenticating user: finding user from
key: 192.168.123.123
[13/Jun/2020:13:03:09][http-9180-Processor24]: FlatFileAuth: User not found in password file.
```

9.3. CMC 身份验证插件

CMC 注册程序可让注册的客户端使用 CMC 身份验证插件进行身份验证，该插件由证书请求在代理证书或用户证书中预签名，具体取决于插件。当收到使用有效证书签名的 CMC 请求时，证书管理器会自动发布证书。

CMC 身份验证插件还为客户端提供 CMC 撤销程序。CMC 撤销程序可让客户端具有由代理证书签名的证书请求，或者验证拥有证书的用户，然后将此类请求发送到证书管理器。当收到使用有效证书签名的 CMC 撤销请求时，证书管理器会自动撤销证书。

证书系统提供以下 CMC 身份验证插件：

CMCAuth

当 CA 代理为 CMC 请求签名时，请使用此插件。

要使用 CMCAuth 插件，请在注册配置集中设置以下内容：

```
auth.instance_id=CMCAuth
```

默认情况下，以下注册配置集使用 CMCAuth 插件：

- 对于系统证书：
  - caCMCauditSigningCert
  - caCMCcaCert
  - caCMCECserverCert
  - caCMCECsubsystemCert
  - caCMCECUserCert
  - caCMCkraStorageCert
  - caCMCkraTransportCert

- caCMCocspCert
- caCMCserverCert
- caCMCsubsystemCert

- 对于用户证书：
  - caCMCUserCert
  - caECFullCMCUserCert
  - caFullCMCUserCert

CMCUserSignedAuth

当用户提交签名或基于 SharedSecret 的 CMC 请求时，请使用此插件。

要使用 CMCUserSignedAuth 插件，请在注册配置集中设置以下内容：

```
auth.instance_id=CMCUserSignedAuth
```

用户签名的 CMC 请求必须由用户的证书签名，该证书包含与请求的证书相同的 subjectDN 属性。如果用户已获取了一个签名证书，则您只能使用用户签名的 CMC 请求，以证明其用于其他证书的身份。

基于 SharedSecret 的 CMC 请求表示请求由请求本身的私钥签名。在这种情况下，CMC 请求必须使用 Shared Secret 机制进行身份验证。基于 SharedSecret 的 CMC 请求通常是用来获取用户的第一个签名证书，稍后用于获取其他证书。详情请查看 第 9.4 节 “CMC SharedSecret 身份验证”。

默认情况下，以下注册配置集使用 CMCUserSignedAuth 插件：

- caFullCMCUserSignedCert
- caECFullCMCUserSignedCert
- caFullCMCSharedTokenCert
- caECFullCMCSharedTokenCert

9.4. CMC SHAREDSECRET 身份验证

使用 Shared Secret 功能可让用户向服务器发送未签名的 CMC 请求。例如，如果用户要获取第一个签名证书，则需要这样做。之后，可以使用此签名证书对这个用户的其他证书进行签名。

9.4.1. 创建共享 secret 令牌

Red Hat Certificate System planning、Installing 和 Deployment Guide 中的 Shared Secret Workflow 部分描述了使用 Shared Secret Token 时的工作流。根据情况，最终用户或管理员创建共享 Secret 令牌。

注意

要使用共享 secret 令牌，证书系统必须使用 RSA 保障证书。详情请参阅 RHCS 规划、安装和部署指南中的启用 CMC 共享 Secret 功能部分。

要创建共享 Secret 令牌，请输入：

```
# CMCSharedToken -d /home/user_name/.dogtag/ -p NSS_password \
    -s "CMC_enrollment_password" -o /home/user_name/CMC_shared_token.b64 \
    -n "issuance_protection_certificate_nickname"
```

如果使用 HSM，则额外将 -h token_name 选项传递给命令，以设置 HSM 安全令牌名称。

有关 CMCSharedToken 工具程序的详情，请查看 CMCSharedToken(8) man page。

注意

生成的令牌已加密，只有生成的用户才知道密码。如果 CA 管理员为用户生成令牌，管理员必须以安全的方式向用户提供密码。

创建 Shared Token 后，管理员必须将令牌添加到用户或证书记录中。详情请查看 第 9.4.2 节 “设置 CMC 共享 Secret”。

9.4.2. 设置 CMC 共享 Secret

根据计划的操作，管理员必须在用户或证书的 LDAP 条目中生成一个 Shared Secret Token。

有关工作流和使用共享 Secret 的详情，请查看 Red Hat 证书系统规划、安装和部署指南中的 " 共享 Secret 工作流 "部分。

9.4.2.1. 将 CMC 共享 Secret 添加到用于证书注册的用户条目

要将 Shared Secret Token 用于证书注册，请将它作为管理员存储在用户的 LDAP 条目中：

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user_name,ou=People,dc=example,dc=com
changetype: modify
replace: shrTok
shrTok: base64-encoded_token
```

9.4.2.2. 将 CMC 共享 Secret 添加到证书撤销的证书

要将 Shared Secret Token 用于证书撤销，请将它作为管理员存储在要撤销的证书的 LDAP 条目中：

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=certificate_id,ou=certificateRepository,ou=ca,o=pki-tomcat-CA
changetype: modify
replace: shrTok
shrTok: base64-encoded_token
```

9.5. 测试注册

有关通过配置集测试注册的详情，请参考 第 3 章 制作发行证书规则（证书配置文件）。测试最终用户是否可以使用验证方法集成功注册证书：

1. 打开"终端"页面。

   ```
   https://server.example.com:8443/ca/ee/ca
   ```

2. 在 Enrollment 选项卡中，打开自定义注册表单。

3. 填写值并提交请求。

4. 提示时输入密钥数据库的密码。

5. 输入正确的密码时，客户端会生成密钥对。

   不要中断密钥进程。在完成密钥生成后，请求将提交到服务器以发布证书。服务器对证书配置文件的请求进行约束，只有在请求满足所有要求时才会发布证书。

   签发证书后，在浏览器中安装证书。

6. 验证证书是否在浏览器的证书数据库中安装。

7. 如果基于 PIN 的目录身份验证配置了 PIN 删除，则使用同一 PIN 为另一个证书重新注册。请求应该被拒绝。

9.6. 注册自定义身份验证插件

自定义身份验证插件模块可以通过 CA 控制台进行注册。身份验证插件模块也可以通过 CA 控制台删除。在删除模块之前，删除基于该模块的实例。

注意

要编写自定义插件，请参阅 身份验证插件。

1. 创建自定义身份验证类。在本例中，自定义身份验证插件名为 **UidPwdDirAuthenticationTestms.java**。

2. 编译新类。

```
javac -d . -classpath $CLASSPATH UidPwdDirAuthenticationTestms.java
```

3. 在 CA 的 WEB-INF web 目录中创建一个目录来存放自定义类，以便 CA 可以访问它们以获取报名表。

```
mkdir /usr/share/pki/ca/webapps/ca/WEB-INF/classes
```

4. 将新插件文件复制到 新类 目录中，并将所有者设置为证书证书系统nbsp;System 系统用户 (pkiuser)。

```
cp -pr com /usr/share/pki/ca/webapps/ca/WEB-INF/classes

chown -R pkiuser:pkiuser /usr/share/pki/ca/webapps/ca/WEB-INF/classes
```

5. 登录到控制台。

```
pkiconsole https://server.example.com:8443/ca
```

6. 注册插件。

   a. 在 Configuration 选项卡中，单击导航树中的 Authentication。

   b. 在右侧窗格中，单击 Authentication Plug-in Registration 选项卡。

选项卡中列出了已经注册的模块。

c.

要注册插件，请点击 **Register**。

此时会出现 **Register Authentication Plug-in Implementation** 窗口。

d.

通过填写以下两个字段来注册的模块：

- 插件名称。模块的名称。
- 类名称。此模块的类全名。这是实现 **Java™** 类的路径。如果这个类是软件包的一部分，请包含软件包名称。例如，要在名为 **com.customplugins** 的软件包中注册名为 **customAuth** 的类，类名称为 **com.customplugins.customAuth**。

7.

在注册该模块后，将模块添加为活跃的身份验证实例。

a.

在 **Configuration** 选项卡中，单击导航树中的 **Authentication**。

b.

在右侧窗格中，单击 **Authentication Instance** 选项卡。

c.

点 **Add**。

d.

从列表中选择自定义模块 **UidPwdDirAuthenticationTestms.java** 以添加模块。填写该模块的适当配置。

8.

创建新的端到端注册表，以使用新的身份验证模块。

```
cd /var/lib/pki/pki-tomcat/ca/profiles/ca

cp -p caDirUserCert.cfg caDirUserCertTestms.cfg

vi caDirUserCertTestms.cfg
```

```
desc=Test ms - This certificate profile is for enrolling user certificates with directory-based
authentication.
visible=true
enable=true
enableBy=admin
name=Test ms - Directory-Authenticated User Dual-Use Certificate Enrollment
auth.instance_id=testms
...
```

9. **将新配置集添加到 CA 的 CS.cfg 文件中。**

提示

编辑前备份 **CS.cfg** 文件。

```
vim /var/lib/pki/instance-name/ca/conf/CS.cfg

profile.list=caUserCert,caDualCert,caSignedLogCert,caTPSCert,caRARouterCert,caRouterCer
t,caServerCert,caOtherCert,caCACert,caInstallCACert,caRACert,caOCSPCert,caTransportCe
rt,caDirUserCert,caAgentServerCert,caAgentFileSigning,caCMCUserCert,caFullCMCUserCert
,caSimpleCMCUserCert,caTokenDeviceKeyEnrollment,caTokenUserEncryptionKeyEnrollment,
caTokenUserSigningKeyEnrollment,caTempTokenDeviceKeyEnrollment,caTempTokenUserEn
cryptionKeyEnrollment,caTempTokenUserSigningKeyEnrollment,caAdminCert,caInternalAuthS
erverCert,caInternalAuthTransportCert,caInternalAuthKRAstorageCert,caInternalAuthSubsyste
mCert,caInternalAuthOCSPCert,DomainController,caDirUserCertTestms
...
profile.caDirUserCertTestms.class_id=caEnrollImpl
profile.caDirUserCertTestms.config=/var/lib/pki/pki-
tomcat/ca/profiles/ca/caDirUserCertTestms.cfg
```

10. **重启 CA。**

```
systemctl restart pki-tomcatd@instance_name.service
```

9.7. 使用命令行手动检查证书状态

要检查证书请求，请确保您已通过正确权限进行身份验证，以批准证书请求。有关配置 pki 命令行界面的详情，请参考 第 2.5.1.1 节 “pki CLI initialization”。

查看请求：

1. 显示待处理的证书请求列表：

```
$ pki agent_authentication_parameters ca-cert-request-find --status pending
```

此命令列出所有待处理的证书请求。

2. 下载特定证书请求：

```
$ pki agent_authentication_parameters ca-cert-request-review id --file request.xml
```

3. 在编辑器或一个单独的终端中打开 request.xml 文件，并查看请求的内容以确保其是合法的。然后，回答提示：如果请求有效，回答"批准 并按 Enter 键。如果请求无效，请回答 reject 并按 Enter。组织可以订阅语义差别 以拒绝 和 取消 ；它们都无法发布证书。

9.8. 使用 WEB 界面手动检查证书状态

1. 在网页浏览器中打开以下 URL：

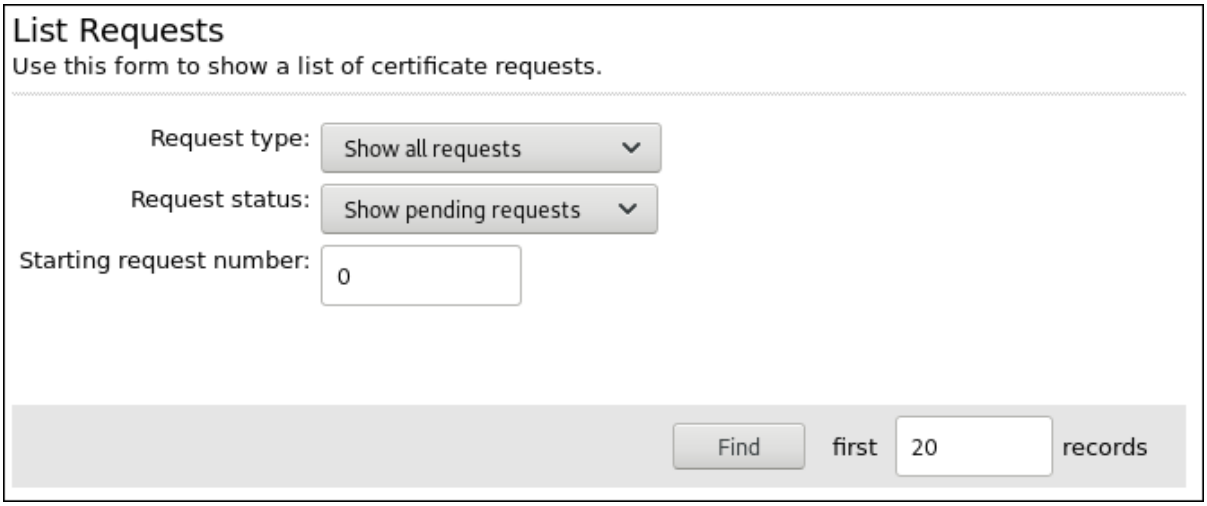
```
https://server_host_name:8443/ca/agent/ca
```

2. 作为代理进行身份验证。有关以用户身份进行身份验证和配置浏览器的详情，请参考 第 2.4.1 节 “浏览器初始化”。

3. 在左侧的边栏中，单击 List requests 链接。

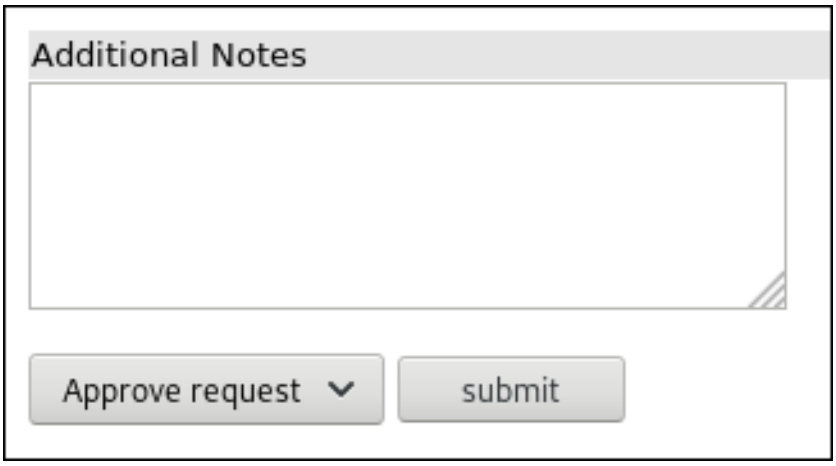
4. 选择 Show all requests for Request type and Show pending requests for Request status 来过滤请求。

5. 点右下角的 Find。



6. *results 页面会列出等待查看的所有待处理请求。点击请求号来查看请求。*

7. *检查请求信息并确保它是合法的请求。如有必要，修改策略信息以更正任何错误，或对证书进行任何更改，例如更改 未有效的 after 字段。（可选）保留额外的备注。*



下拉菜单包括几个检查状态更新。选择 Approve request 以批准批准请求或拒绝请求以拒绝它，然后单击 Submit。机构可以订阅语义差别，以拒绝请求和 取消请求 ；它们都没有签发证书。

第 10 章 注册证书授权（访问 EVALUATORS）

本章论述了使用访问评估器的授权机制。

10.1. 授权机制

除了身份验证机制外，每个注册的配置文件也可以配置为具有自己的 授权机制。授权机制仅在成功身份验证后执行。

授权机制由 Access Evaluator 插件框架提供。访问评估器是用于评估访问控制指令(ACI)条目的可插入类。机制提供了一种评估方法，它采用预定义的参数列表（即 类型、op、值），评估表达式，如 group='Certificate Manager Agents'，并根据评估结果返回布尔值。

10.2. 默认评估器

Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System 提供 4 个默认评估器。在 CS.cfg 文件中会默认列出以下条目：

```
accessEvaluator.impl.group.class=com.netscape.cms.evaluators.GroupAccessEvaluator
accessEvaluator.impl.ipaddress.class=com.netscape.cms.evaluators.IPAddressAccessEvaluator
accessEvaluator.impl.user.class=com.netscape.cms.evaluators.UserAccessEvaluator
accessEvaluator.impl.user_origreq.class=com.netscape.cms.evaluators.UserOrigReqAccessEvaluator
```

组 访问评估器评估用户的组成员资格属性。例如，在以下注册配置集条目中，只有 CA 代理才可以使用该配置集进行注册：

```
authz.acl=group="Certificate Manager Agents"
```

ipaddress access evaluator 评估请求主体的 IP 地址。例如，在以下注册配置集条目中，只有指定 IP 地址的主机才可以使用该配置集进行注册：

```
authz.acl=ipaddress="a.b.c.d.e.f"
```

用户访问 evaluator 评估用户 ID 完全匹配。例如，在以下注册配置集条目中，只有与列出的用户匹配的用户才可以使用该配置集进行注册：

```
authz.acl=user="bob"
```

user_origreq 访问 evaluator 根据前一个匹配的请求评估经过身份验证的用户。此特殊评估器专门设计用于续订的用户，确保请求续订的用户是拥有原始请求的用户。例如，在以下续订注册配置集条目中，经过身份验证的用户的 UID 必须与请求续订的用户 UID 匹配：

```
authz.acl=user_origreq="auth_token.uid"
```

新的评估器可以在当前框架中编写，并可通过 CS 控制台进行注册。默认的 evaluators 可用作模板，以展开和自定义到更目标的插件。

第 11 章 使用自动通知

CertificateCertificate Systemnbsp;System 可以配置为在签发或撤销证书时向最终用户发送自动电子邮件通知，或者在新请求到达代理请求队列中时向代理发送。本章论述了自动通知和详情如何启用、配置和自定义发送的通知电子邮件消息。

注意

自动 通知 不会与自动化作业混淆。有关此主题的更多信息，请参阅 第 12 章 设置自动化任务。

注意

由于可以发送的通知类型，只有证书管理器才能够为通知配置；此选项不可用于其他子系统。

11.1. 关于 CA 自动通知

自动化通知是发生指定事件时发送的电子邮件消息。系统使用监控系统的监听程序来决定特定事件发生的时间；事件发生时，会触发系统向配置的接收者发送电子邮件。每种通知都使用纯文本或 HTML 模板来构造通知消息。模板包含文本和令牌，被扩展以填充特定事件的正确信息。可以通过更改模板中包含的文本和令牌来自定义消息。也可以针对不同的外观和格式自定义 HTML 模板。

11.1.1. 自动通知的类型

有三种自动通知类型：

- 证书颁发的.

  通知消息会自动发送给已签发证书的用户。如果用户的证书请求被拒绝，则会向用户发送拒绝消息。

- 证书撤销.

  在撤销用户证书时，会自动向用户发送通知消息。

- 在 Queue 中请求。

  当请求使用为代理设置的电子邮件地址时，当请求进入代理时，通知消息会自动发送到一个或多个代理。此通知类型在每次消息进入队列时发送电子邮件。有关队列作业中请求的更多信息，请参阅 第 12.1.2.2 节 “requestInQueueNotifier (RequestInQueueJob)”。

  还有一个作业，将通知发送到队列状态的代理，其中包括队列状态在特定时间段内的摘要。

11.1.2. 确定最终用户地址

通知系统首先检查证书请求或撤销请求来确定端点的电子邮件地址，然后检查证书的主题名称，如果证书包含此扩展，则会确定证书的 Subject Alternative Name 扩展。如果无法找到电子邮件地址，则将通知发送到通知面板的 Sender 的 Email Address 字段中指定的电子邮件地址。

11.2. 为 CA 设置自动通知

11.2.1. 在控制台中设置自动通知

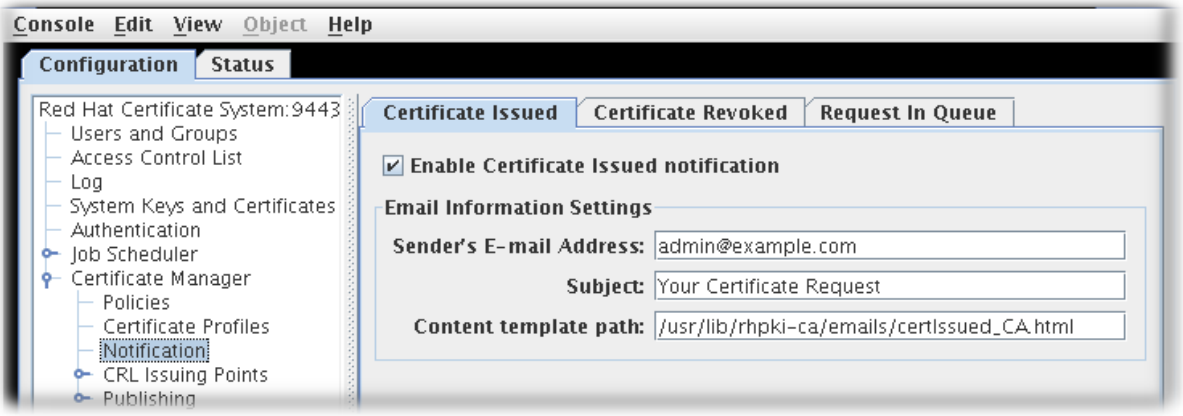
1. 打开证书管理器控制台。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. 打开 Configuration 选项卡。

3. 在左侧的导航树中打开 Certificate Manager 标题。然后选择 通知。

   Notification 标签页会出现在窗口的右侧。



4. *可以为三种事件发送通知：新发布的证书、撤销证书和新证书请求。要为任何事件发送通知，选择选项卡，选中"启用"复选框，并在以下字段中指定信息：*

   - *发件人的电子邮件地址 .键入收到发送问题的用户的发件人完整电子邮件地址。*
   - *收件人的电子邮件地址 .这些是将检查队列的代理的电子邮件地址。要列出多个接收者，请使用逗号将电子邮件地址分开。仅适用于队列中的新请求。*
   - *主题.键入通知的主题标题。*
   - *内容模板路径.键入路径，包括文件名，到包含模板用来构造消息内容的目录。*

5. *点 Save。*

   注意

   确保邮件服务器设置正确。请参阅 第 11.4 节 “为证书证书系统配置邮件服务器; 系统通知”。

6. *自定义通知消息模板。如需更多信息，请参阅 第 11.3 节 “自定义通知消息”。*

7. *测试配置。请参阅 第 11.2.3 节 “测试配置”。*

11.2.2. 通过编辑 CS.cfg 文件配置特定通知

1. 停止 CA 子系统。

```
systemctl stop pki-tomcatd@instance_name.service
```

2. 打开该实例的 CS.cfg 文件。此文件位于实例的 conf/ 目录中。

3. 编辑正在启用的通知类型的所有配置参数。

对于证书发出通知，有四个参数：

```
ca.notification.certIssued.emailSubject
ca.notification.certIssued.emailTemplate
ca.notification.certIssued.enabled
ca.notification.certIssued.senderEmail
```

对于证书撤销通知，有四个参数：

```
ca.notification.certRevoked.emailSubject
ca.notification.certRevoked.emailTemplate
ca.notification.certRevoked.enabled
ca.notification.certRevoked.senderEmail
```

对于证书请求通知，有五个参数：

```
ca.notification.requestInQ.emailSubject
ca.notification.requestInQ.emailTemplate
ca.notification.requestInQ.enabled
ca.notification.requestInQ.recipientEmail
ca.notification.requestInQ.senderEmail
```

通知消息的参数在 第 11.2 节 “为 CA 设置自动通知” 中解释。

4. 保存该文件。

5. 重启 CA 实例。

```
systemctl start pki-tomcatd@instance_name.service
```

6. 如果创建了作业来发送自动化消息，请检查邮件服务器是否已正确配置。请参阅 第 11.4 节 “为证书证书系统配置邮件服务器; 系统通知”。

7. 自动发送的消息可以自定义；如需更多信息，请参阅 第 11.3 节 “自定义通知消息”。

11.2.3. 测试配置

要测试子系统是否按配置发送电子邮件通知，请执行以下操作：

1. 将队列通知中请求的通知电子邮件地址更改为可访问代理或管理员电子邮件地址。

2. 打开终端页面，并使用代理批准的报名表请求证书。

   当请求排队进行代理批准时，应该发送 **request-in-queue** 电子邮件通知。检查消息以查看它是否包含配置信息。

3. 登录到代理接口，并批准请求。

   当服务器发布证书时，用户会向请求中列出的地址收到证书签发的电子邮件通知。检查消息以查看它是否具有正确的信息。

4. 登录到代理接口，并撤销证书。

   用户电子邮件帐户应包含一条电子邮件消息读取证书已被撤销。检查消息以查看它是否具有正确的信息。

11.3. 自定义通知消息

电子邮件通知使用每种消息类型的模板进行构建。这样，信息可以易于可重复使用，并方便自定义。**CA** 使用模板作为其通知消息。存在独立的模板，适用于 **HTML** 和纯文本消息。

11.3.1. 自定义 CA 通知消息

每种 CA 通知消息都有一个 HTML 模板，并关联一个纯文本模板。消息构建自文本、令牌，以及 HTML 模板 HTML 标记。令牌是 变量，由消息中的美元符号($)标识，在消息被构造时替换为当前值。有关可用令牌列表，请参阅 表 11.3 “通知变量”。

可以通过更改消息模板中的文本和令牌来修改消息类型的内容。可以通过修改 HTML 消息模板中的 HTML 命令来更改 HTML 消息的外观。

证书验证信息的默认文本版本如下：

```
Your certificate request has been processed successfully.
SubjectDN= $SubjectDN
IssuerDN= $IssuerDN
notAfter= $NotAfter
notBefore= $NotBefore
Serial Number= 0x$HexSerialNumber
To get your certificate, please follow this URL:
https://$HttpHost:$HttpPort/displayBySerial?op=displayBySerial&
 serialNumber=$SerialNumber
Please contact your admin if there is any problem.
And, of course, this is just a \$SAMPLE\$ email notification form.
```

通过重新排列、添加或删除令牌和文本，可以根据需要自定义此模板，如下所示：

```
THE EXAMPLE COMPANY CERTIFICATE ISSUANCE CENTER
Your certificate has been issued!
You can pick up your new certificate at the following website:
https://$HttpHost:$HttpPort/displayBySerial?op=displayBySerial&
 serialNumber=$SerialNumber
This certificate has been issued with the following information:
Serial Number= 0x$HexSerialNumber
Name of Certificate Holder = $SubjectDN
Name of Issuer = $IssuerDN
Certificate Expiration Date = $NotAfter
Certificate Validity Date = $NotBefore
Contact IT by calling X1234, or going to the IT website http://IT
 if you have any problems.
```

通知模板位于 /var/lib/pki/instance_name/ca/emails 目录中。

可以更改这些消息的名称和位置；在配置通知时进行适当的更改。所有模板名称都可以更改，除了证书被拒绝的模板外，这些名称必须保持相同。与证书关联的模板需要位于同一目录中，且证书拒绝的模板

必须位于同一目录中，并且必须使用相同的扩展名。

表 11.1 “通知模板” 列出为创建通知消息提供的默认模板文件。表 11.2 “作业通知电子邮件模板” 列出为创建作业摘要信息而提供的默认模板文件。

表 11.1. 通知模板

文件名	描述
certIssued_CA	发布证书时，纯文本通知电子邮件模板到结束实体。
certIssued_CA.html	签发证书时，基于 HTML 的通知电子邮件模板以结束实体。
certRequestRejected.html	当证书请求被拒绝时，基于 HTML 的通知电子邮件模板到最终实体。
certRequestRevoked_CA	撤销证书时，用于纯文本通知电子邮件到结束实体的模板。
certRequestRevoked_CA.html	撤销证书时，基于 HTML 的通知电子邮件模板以结束实体。
reqInQueue_CA	当请求进入队列时，向代理发出纯文本通知电子邮件的模板。
reqInQueue_CA.html	当请求进入队列时，适用于基于 HTML 的通知电子邮件模板到代理。

表 11.2. 作业通知电子邮件模板

文件名	描述
rnJob1.txt	用于创建发送到最终用户的消息内容的模板，告知他们其证书即将过期，并在证书过期之前被续订或替换。
rnJob1Summary.txt	构建要发送到代理和管理员的摘要报告的模板。使用 **rnJob1Item.txt** 模板来格式化消息中的项目。
rnJob1Item.txt	用于格式化摘要报告中包含的项目的模板。
riq1Item.html	用于格式化摘要表中所含项目的模板，该模板使用 **riq1Summary.html** 模板进行构建。
riq1Summary.html	用于公式表示报告或表的模板，用于概述证书管理器的代理队列中的待处理请求数量。

文件名	描述
publishCerts	报告或表的模板，用于汇总要发布到该目录的证书。使用 **publishCertsItem.html** 模板格式化表中的项目。
publishCertsItem.html	用于格式化摘要表中所含项目的模板。
ExpiredUnpublishJob	汇总从目录中过期的证书的报告或表的模板。使用 **ExpiredUnpublishJobItem** 模板来格式化表中的项目。
ExpiredUnpublishJobItem	用于格式化摘要表中所含项目的模板。

表 11.3 “通知变量” 列出并定义可在通知模板中使用的变量。

表 11.3. 通知变量

令牌	描述
$CertType	指定证书的类型，可以是以下任意一种： • TLS 客户端（**客户端**） • TLS 服务器（**服务器**） • CA 签名证书(**ca**) • 其他（**其他**）。
$ExecutionTime	给出作业运行的时间。
$HexSerialNumber	为以十六进制格式发布的证书指定序列号。
$HttpHost	授予证书管理器的完全限定主机名，以便最终实体应连接到其证书以检索其证书。
$HttpPort	给出证书管理器的结束日期（非 TLS）端口号。
$InstanceID	提供发送通知的子系统的 ID。
$IssuerDN	提供签发证书的 CA 的 DN。
$NotAfter	提供有效期日期。
$NotBefore	给出有效期限的开始日期。
$RecipientEmail	给出接收者的电子邮件地址。

令牌	描述
$RequestId	给出请求 ID。
$RequestorEmail	给出请求者的电子邮件地址。
$RequestType	提出的请求类型。
$RevocationDate	给出证书被撤销的日期。
$SenderEmail	给出发件人的电子邮件地址；这与通知配置中的 **Sender 的 E-mail Address 字段中指定的地址** 相同。
$SerialNumber	为发出的证书指定序列号 ; 序列号显示为生成的消息中的十六进制值。
$Status	请求状态。
$SubjectDN	提供证书主体的 DN。
$SummaryItemList	列出摘要通知中的项目。每个项目对应作业检测或从发布目录中删除的证书。
$SummaryTotalFailure	指定摘要报告失败的项目总数。
$SummaryTotalNum	指定在队列中待处理的证书请求总数或要从概述报告中的目录中更新或删除的证书总数。
$SummaryTotalSuccess	显示摘要报告中项目总数的数目。

11.4. 为证书证书系统配置邮件服务器; 系统通知

通知和作业功能使用在 CertificateCertificate Systemnbsp;System CA 实例中发送通知邮件中配置的邮件服务器。通过执行以下操作设置邮件服务器：

1. *打开 CA 子系统管理控制台。例如：*

```
pkiconsole https://server.example.com:8443/ca
```

2. *在配置 选项卡中，突出显示顶部的实例名称，然后选择 SMTP 选项卡。*

3.
提供邮件服务器的服务器名称和端口号。

服务器名称是安装邮件服务器的计算机的完全限定 DNS 主机名，如 mail.example.com。默认情况下，邮件服务器的主机名为 localhost，而不是实际主机名。

SMTP 邮件服务器侦听的默认端口号为 25。

4.
点 Save。

11.5. 为 CA 创建自定义通知

通过为证书Certificate Systemnbsp 编辑现有的电子邮件通知插件，可以创建自定义通知功能来处理其他 PKI 操作，如令牌注册。在尝试创建和使用自定义通知插件之前，请联系红帽nbsp;Hat 支持服务。

第 12 章 设置自动化任务

CertificateCertificate Systemnbsp;System 提供了一个自定义作业调度程序，它支持各种用于调度 cron 任务的机制。本章论述了如何配置 CertificateCertificate Systemnbsp;System 以使用特定作业插件模块来完成作业。

注意

自动化作业 不会与自动 通知 混淆。有关此主题的更多信息，请参阅 第 11 章 使用自动通知。

12.1. 关于自动任务

证书管理器控制台包含一个 作业调度程序选项，可以在指定时间执行特定的作业。Job Scheduler 与传统的 Unix cron 守护进程类似；它使用注册的 cron 作业，并在预先配置的日期和时间执行它们。如果配置，调度程序会检查等待执行的作业的指定间隔；如果指定的执行时间已到达，调度程序会自动启动该作业。

作业使用 Java™ 类实施，然后使用 CertificateCertificate Systemnbsp 进行了注册；System as plug-in 模块。可以使用 job 模块的一个实施来配置作业的多个实例。每个实例必须具有唯一的名称（无空格的字母数字字符串），并可包含不同的输入参数值以应用到不同的作业。

12.1.1. 设置自动化任务

自动化作业功能通过执行以下操作来设置：

- 启用和配置作业调度程序；如需更多信息，请参阅 第 12.2 节 “设置作业调度程序”。
- 为这些作业模块启用和配置作业模块并设置首选项；如需更多信息，请参阅 第 12.3 节 “设置特定作业”。
- 通过更改与通知类型关联的模板，自定义这些作业发送的电子邮件通知消息。消息内容由纯文本消息和 HTML 消息组成；通过更改 HTML 模板来修改外观。如需更多信息，请参阅 第 11.3.1 节 “自定义 CA 通知消息”。

12.1.2. Automated 任务类型

自动作业类型是 RenewalNotificationJob、RequestInQueueJob、PublishCertsJob 和 UnpublishExpiredJob。在部署证书系统时，会创建每种作业类型的一个实例。

12.1.2.1. certRenewalNotifier (RenewalNotificationJob)

certRenewalNotifier 作业会检查即将在内部数据库中过期的证书。找到一个时，它会自动向证书的所有者发送电子邮件，并在配置的时间或直到证书被替换前继续发送电子邮件提醒。该作业将收集所有续订通知的汇总信息，并将摘要发送到配置的代理或管理员。

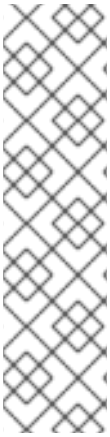
作业使用电子邮件解析器确定要发送通知的电子邮件地址。默认情况下，电子邮件地址在证书本身或证书相关的注册请求中找到。

12.1.2.2. requestInQueueNotifier (RequestInQueueJob)

requestInQueueNotifier 作业以预先配置的时间间隔检查请求队列的状态。如果队列中等待任何延迟的注册请求，作业会构建一封电子邮件消息，信息汇总其结果并将其发送到指定的代理。

12.1.2.3. publishCerts (PublishCertsJob)

发布 证书作业会检查添加到已发布的发布目录中的所有新证书。当添加了这些新证书时，通过发布 Certs 作业会自动将它们发布到 LDAP 目录或文件中。



注意

大多数时候，发布者会立即发布任何与规则匹配的证书，并将其规则与相应的发布目录一起发布。

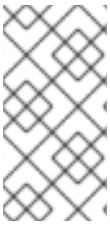
如果在创建证书时成功发布证书，则 publishCerts 作业将不会重新发布该证书。因此，新证书不会在作业摘要报告中列出，因为摘要只会列出 publishCerts 作业发布的证书。

12.1.2.4. unpublishExpiredCerts (UnpublishExpiredJob)

过期的证书不会自动从发布目录中删除。如果证书管理器被配置为将证书发布到 LDAP 目录，则 目录会包含过期的证书。

unpublishExpiredCerts 作业检查是否已过期且仍然标记为已配置的时间间隔在内部数据库中 发布的 证书。作业连接到发布目录并删除这些证书，然后将这些证书标记为内部数据库中 未发布。作业收集

其删除的过期证书的摘要，并将摘要发送到配置指定的代理或管理员。



注意

此作业自动从目录中删除过期的证书。过期的证书也可以手动删除。有关这个证书的更多信息，请参阅 第 8.12 节 “更新目录中的证书和 CRL”。

12.2. 设置作业调度程序

只有在启用了作业调度程序时，证书管理器才能执行作业。作业设置（如启用作业调度、设置频率和启用作业模块）可以通过证书Certificate Systemnbsp;System CA Console 或编辑 CS.cfg 文件来完成。

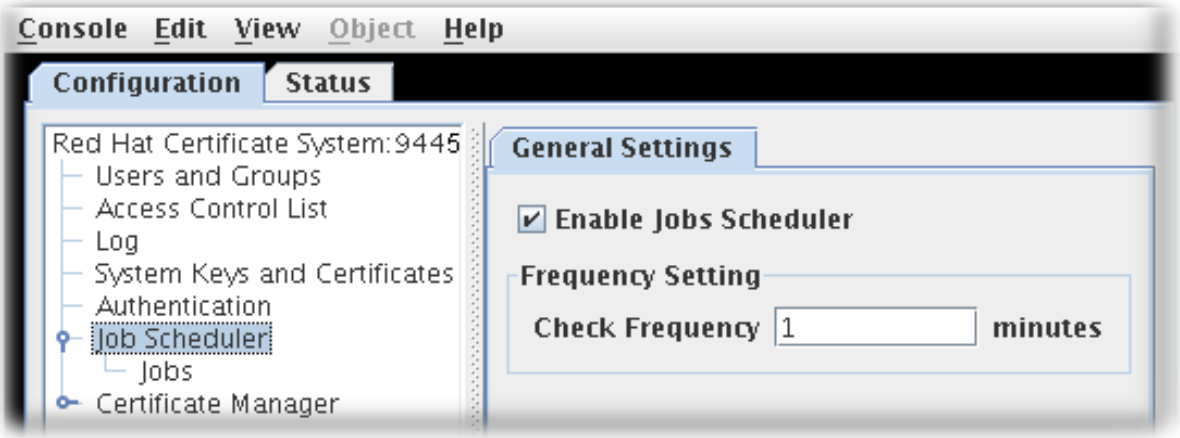
打开 Job Scheduler：

1. 打开证书管理器控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡导航树中，单击 Job Scheduler。

这将打开 General Settings 选项卡，其中显示作业调度程序当前是否已启用。



3. 单击 Enable Jobs Schedule 复选框，以启用或禁用作业调度程序。

禁用作业调度程序会关闭所有作业。

4.
设置调度程序在 Check Frequency 字段中检查作业的频率。

频率是作业调度程序守护进程线程的启动和调用符合 cron 规格的已配置作业的频率。默认情况下，它被设置为一分钟。

注意

输入此信息的窗口可能太小以查看输入信息。拖动证书管理器控制台的基石，以扩大整个窗口。

5.
点 Save。

12.3. 设置特定作业

可以通过证书管理器控制台或编辑配置文件目录来配置自动化作业。建议通过证书管理器控制台进行这些更改。

12.3.1. 使用证书管理器控制台配置特定作业

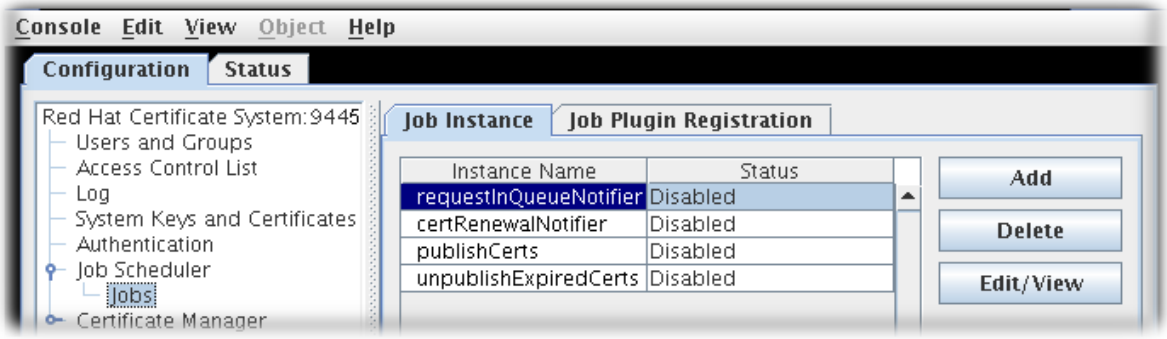
使用证书管理器控制台启用和配置自动化作业：

1.
打开证书管理器控制台。

```
pkiconsole https://server.example.com:8443/ca
```

2.
确认作业调度程序已启用。如需更多信息，请参阅 第 12.2 节 “设置作业调度程序”。

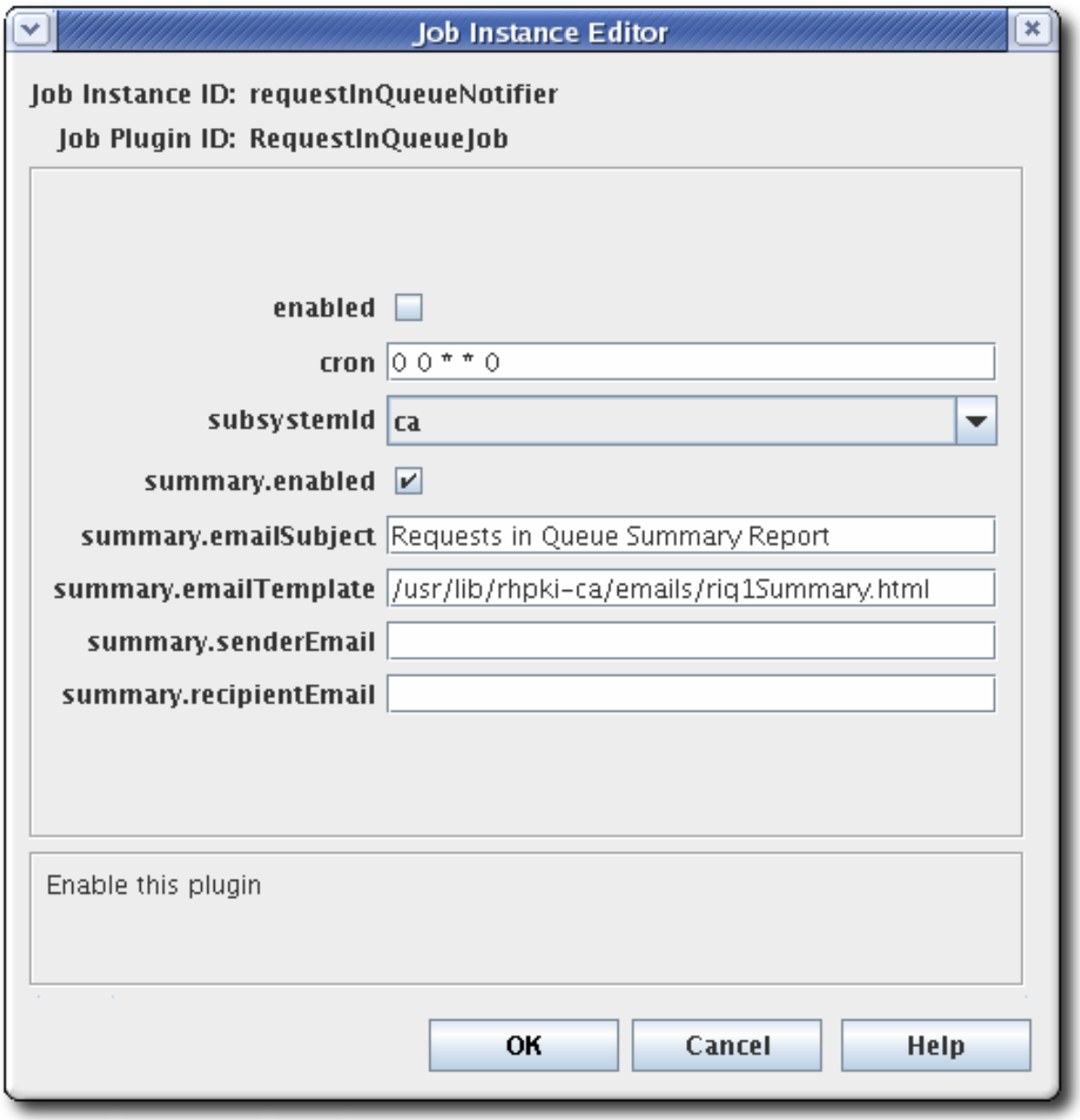
3.
在 Configuration 选项卡中，从导航树中选择 Job Scheduler。然后选择 Jobs 以打开 Job Instance 选项卡。



*从列表中选择作业实例，再单击 **Edit/View**。*

Job Instance Editor 打开显示当前作业配置。

图 12.1. 作业配置



4. 选择 **enabled** 以打开作业。

5. 通过在对话框的字段中指定配置设置来设置配置设置。

   - 有关 **certRenewalNotifier**，请参阅 第 12.3.3 节 “certRenewalNotifier 的配置参数”。
   - 对于 **requestInQueueNotifier**，请参阅 第 12.3.4 节 “requestInQueueNotifier 的配置参数”。
   - 有关 发布证书，请参阅 第 12.3.5 节 “发布证书的配置参数”。
   - 有关 **unpublishExpiredCerts**，请参阅 第 12.3.6 节 “unpublishExpiredCerts 的配置参数”。
   - 有关设置 **cron** 时间频率的详情，请参考 第 12.3.7 节 “自动任务的频率设置”。

6. 点击 **OK**。

7. 点 **Refresh** 查看主窗口中的任何更改。

8. 如果作业被配置为发送自动消息，请检查邮件服务器是否设置正确。请参阅 第 11.4 节 “为证书证书系统配置邮件服务器; 系统通知”。

9. 自定义电子邮件消息文本和外观。

12.3.2. 通过编辑配置文件配置作业

1. 确保已启用并配置了 **Jobs Scheduler**；请参阅 第 12.2 节 “设置作业调度程序”。

2. 停止 CA 子系统实例。

```
systemctl stop pki-tomcatd@instance_name.service
```

3. 在文本编辑器中打开该服务器实例的 CS.cfg 文件。

4. 编辑正在配置的作业模块的所有配置参数。

- 要配置 certRenewalNotifier 作业，请编辑以 作业 Scheduler.job.certRenewalNotifier 开头的所有参数；请参阅 第 12.3.3 节 “certRenewalNotifier 的配置参数”。
- 要配置 requestInQueueNotifier 作业，请编辑以 作业 Scheduler.job.requestInQueueNotifier 开始的所有参数；请参阅 第 12.3.4 节 “requestInQueueNotifier 的配置参数”。
- 要配置发布证书作业，请编辑以 job Scheduler.job. publishCerts 开头的所有参数; 请参阅 第 12.3.5 节 “发布证书的配置参数”。
- 要配置 未发布ExpiredCerts 作业，请编辑以 job Scheduler.job.unpublishExpiredCerts 开头的所有参数；请参阅 第 12.3.6 节 “unpublishExpiredCerts 的配置参数”。

5. 保存该文件。

6. 重启服务器实例。

```
systemctl start pki-tomcatd@instance_name.service
```

7. 如果作业将发送自动消息，请检查邮件服务器是否设置正确。请参阅 第 11.4 节 “为证书证书系统配置邮件服务器; 系统通知”。

8. 自定义自动作业消息。

12.3.3. certRenewalNotifier 的配置参数

表 12.1 "certRenewalNotifier 参数" 提供可为 certRenewalNotifier 作业配置的、可在 CS.cfg 文件中或证书管理器控制台中配置的每个参数的详细信息。

表 12.1. certRenewalNotifier 参数

参数	描述
enabled	指定作业是启用或禁用的。值 **true** 启用作业；**false** 可禁用它。
cron	设置应运行此作业时的时间表。这将设置作业调度程序守护进程线程检查发送续订通知的证书的时间。这些设置必须遵循 第 12.3.7 节 "自动任务的频率设置" 中的约定。例如： `0 3 * * 1-5` 示例中的作业会在周一到下午 3:00 运行。
notifyTriggerOffset	设定第一次发送通知的证书过期日期前的时长（以天为单位）。
notifyEndOffset	设定证书过期后（以天为单位）在证书没有替换的情况下，继续发送通知的时间。
senderEmail	设置通知消息的发送者，该发件人将收到任何发送问题的通知。
emailSubject	设置通知消息的主题行文本。
emailTemplate	将路径（包括 文件名）设置为包含模板用来创建消息内容的目录。
summary.enabled	设定续订通知的摘要报告应编译和发送。值 **true** 启用发送概述; **false** 可禁用它。如果启用，请设置剩余的摘要参数；服务器需要它们来发送摘要报告。
summary.recipientEmail	指定概述信息的接收者。这些可以是需要知道用户证书或其他用户状态的代理。通过使用逗号分隔每个电子邮件地址来设定多个接收者。
summary.senderEmail	指定摘要消息的发件人电子邮件地址。
summary.emailSubject	提供摘要消息的主题行。
summary.itemTemplate	将路径（包括 文件名）指定到包含模板的目录，用于创建概述报告中各个项目的内容和格式。

参数	描述
summary.emailTemplate	将路径（包括 文件名）指定到包含用于创建摘要报告电子邮件通知的 目录。

12.3.4. requestInQueueNotifier 的配置参数

表 12.2 “requestInQueueNotifier 参数” 提供可为 requestInQueueNotifier 作业配置的、可在 CS.cfg 文件或证书管理器控制台中配置的每个参数的详细信息。

表 12.2. requestInQueueNotifier 参数

参数	描述
enabled	设定作业是否已启用(**true**)还是禁用(**false**)。
cron	设置作业应运行时的时间表。这是作业调度程序守护进程线程检查待处理请求的队列的时间。这个设置必须遵循 第 12.3.7 节 “自动任务的频率设置” 中的约定。例如： `0 0 * * 0`
subsystemid	指定运行作业的子系统。对于证书管理器，唯一可能的值是 **ca**。
summary.enabled	指定作业完成的摘要应该被编译和发送。值 **true** 启用摘要报告； **false** 可禁用它们。如果启用，请设置剩余的摘要参数；服务器需要它们来发送摘要报告。
summary.emailSubject	设置摘要消息的主题行。
summary.emailTemplate	指定包含用于创建摘要报告的模板的目录的路径，包括文件名。
summary.senderEmail	指定通知消息的发送者，该发件人将收到任何发送问题的通知。
summary.recipientEmail	指定概述信息的接收者。这些可以是需要处理待处理请求或其他用户的代理。通过使用逗号分隔每个电子邮件地址来列出多个接收者。

12.3.5. 发布证书的配置参数

表 12.3 “publishCerts Parameters” 提供可为 发布Certs 作业配置的每个参数的详细信息，可以在 CS.cfg 文件中或证书管理器控制台中进行配置。

表 12.3. publishCerts Parameters

参数	描述
enabled	设置作业是否启用。启用 **true** 的值为 true；禁用 **false**。
cron	设置作业运行的时间调度。这是作业调度程序守护进程线程检查证书以从发布目录中删除过期的证书的时间。这个设置必须遵循 第 12.3.7 节 "自动任务的频率设置" 中的约定。例如： `0 0 * * 6`
summary.enabled	指定作业发布的证书的摘要应该被编译并发送。value **true** 启用摘要;**false** 可禁用它们。如果启用，请设置剩余的摘要参数；服务器需要它们来发送摘要报告。
summary.emailSubject	提供摘要消息的主题行。
summary.emailTemplate	指定包含用于创建摘要报告的模板的目录的路径，包括文件名。
summary.itemTemplate	指定包含模板的目录的路径，包括文件名，用于创建摘要报告收集的每个项目的内容和格式。
summary.senderEmail	指定摘要消息的发送者，该发件人将收到任何发送问题的通知。
summary.recipientEmail	指定概述信息的接收者。这些可以是需要知道用户证书或其他用户状态的代理。可以通过使用逗号分隔每个电子邮件地址来设置多个接收者。

12.3.6. unpublishExpiredCerts 的配置参数

表 12.4 "unpublishExpiredCerts Parameters" 提供可为 未发布ExpiresCerts 作业（可在 CS.cfg 文件中或证书管理器控制台）配置的每个参数的详细信息。

表 12.4. unpublishExpiredCerts Parameters

参数	描述
enabled	设置作业是否启用。启用 **true** 的值为 true；禁用 **false**。

参数	描述
cron	设置作业运行的时间调度。这是作业调度程序守护进程线程检查证书以从发布目录中删除过期的证书的时间。这个设置必须遵循 第 12.3.7 节 "自动任务的频率设置" 中的约定。例如： `0 0 * * 6`
summary.enabled	指定作业发布的证书的摘要应该被编译并发送。value **true** 启用摘要;**false** 可禁用它们。如果启用，请设置剩余的摘要参数；服务器需要它们来发送摘要报告。
summary.emailSubject	提供摘要消息的主题行。
summary.emailTemplate	指定包含用于创建摘要报告的模板的目录的路径，包括文件名。
summary.itemTemplate	指定包含模板的目录的路径，包括文件名，用于创建摘要报告收集的每个项目的内容和格式。
summary.senderEmail	指定摘要消息的发送者，该发件人将收到任何发送问题的通知。
summary.recipientEmail	指定概述信息的接收者。这些可以是需要知道用户证书或其他用户状态的代理。可以通过使用逗号分隔每个电子邮件地址来设置多个接收者。

12.3.7. 自动任务的频率设置

Job Scheduler 使用 Unix crontab 条目格式的变体来指定检查作业队列和执行作业的日期和时间。如 表 12.5 "调度任务的时间值" 和 图 12.1 "作业配置" 所示，时间条目格式由五个字段组成。（作业调度程序不使用为 Unix crontab 指定的第六个字段。） 值用空格或标签页分开。

每个字段可以包含单个整数或一组整数，用连字符(-)分隔来代表一个包含范围。要指定所有法律值，字段可以包含一个星号，而不是整数。day 字段可以包含以逗号分隔的值列表。此表达式的语法是

```
Minute Hour Day_of_month Month_of_year Day_of_week
```

表 12.5. 调度任务的时间值

字段	值
minute	0-59

字段	值
hour	0-23
几号	1-31
年月	1-12
周几	0-6 (where 0=Sunday)

例如，以下时间条目每隔 15 分钟指定每小时（1:15、2:15、3:15 等等）：

```
15 * * * *
```

以下示例设置了在 4 月 12 日没有运行的作业：

```
0 12 12 4 *
```

Day-of-month 和 day-of-week 选项可以包含以逗号分隔的值列表，用于指定多个一天。如果指定了两天字段，则规格为 included；即，月份的当天不需要在一周的某一天生效。例如，以下条目指定每个月第一个和每月一开始的午夜的作业执行时间：

```
0 0 1,15 * 1
```

要指定没有其他一天类型的 1 天，请在其他当日字段中使用星号。例如，以下条目在每天下午 3:15 运行作业：

```
15 3 * * 1-5
```

12.4. 注册一个 JOB 模块

可以通过证书管理器控制台注册自定义作业插件。注册新模块涉及指定模块的名称和实施该模块的 Java™ 类的全名。

注册新的作业模块：

1. 创建自定义作业类。在本例中，自定义作业插件名为 MyJob.java。

2. 编译新类。

```
javac -d . -classpath $CLASSPATH MyJob.java
```

3. 在 CA 的 WEB-INF Web 目录中创建用于存放自定义类的目录，以便 CA 可以访问它们。

```
mkdir /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

4. 将新插件文件复制到 新类 目录中，并将所有者设置为证书证书系统nbsp;System 系统用户 (pkiuser)。

```
cp -pr com /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes

chown -R pkiuser:pkiuser /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

5. 注册插件。

   a. 登录到证书管理器控制台。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

   b. 在 Configuration 选项卡中，选择左侧导航树中的 Job Scheduler。选择 Jobs。

   Job Instance 标签页会打开，它会列出所有当前配置的作业。选择 Job Plugin Registration 选项卡。

   c. 单击 Register 以添加新模块。

   d. 在 Register Job Scheduler Plugin Implementation 窗口中，提供以下信息：

   - 插件名称。输入插件模块的名称。
   - 类名称。输入此模块类的全名；这是实施 Java™ 类的路径。如果这个类是软件包

的一部分，请包含软件包名称。例如，若要在名为 com.customplugins 的软件包中注册一个名为 customJob 的类，请键入 com.customplugins.customJob。

e. 点击 OK。

注意

也可以删除作业模块，但不推荐这样做。

如果需要删除模块，请在注册新模块时打开 Job Plugin Registration 选项卡，选择要删除的模块，然后单击 Delete。出现提示时，请确认删除。

部分 IV. 管理 SUBSYSTEM 实例

第 13 章 基本子系统管理

本章论述了证书证书系统空间；系统管理控制台、配置文件和其他基本管理任务，如启动和停止服务器、管理日志、更改端口分配和更改内部数据库。

13.1. PKI 实例

此版本的证书证书系统nbsp；系统继续支持所有子系统 的独立 PKI 实例。

独立的 PKI 实例

- 作为单一基于 Java 的 Apache Tomcat 实例运行，
- 包含一个 PKI 子系统（CA、KRA、OCSP、 TKS 或 TP）以及
- 如果在同一物理机器或虚拟机(VM)上共存，则必须使用唯一端口。

此外，证书证书系统nbsp 的这个版本引入了 共享 PKI 实例 的概念。

共享 PKI 实例

- 作为单一基于 Java 的 Apache Tomcat 实例运行，
- 可以包含与单独的 PKI 实例相同的 PKI 子系统，
- 可以包含最多一种 PKI 子系统的任意组合：
  - CA
  - TKS

- CA, KRA
- CA, OCSP
- TKS, TPS
- CA、KRA、TKS、TPS
- CA, KRA, OCSP, TKS, TPS
- 以此类推。

- 允许该实例中包含的所有子系统共享相同的端口，以及
- 如果多个端口在同一物理机器或虚拟机上在一起，则必须利用唯一端口。

13.2. PKI 实例执行管理

启动、停止、重新启动或获取 PKI 实例的状态的操作称为执行管理。每个 PKI 实例（独立或共享）都是启动、停止、重启和分别获取其状态。本节介绍任何 PKI 实例的执行管理。

13.2.1. 启动、停止和重启 PKI 实例

PKI 实例被启动、停止和重启，就像使用 systemd 的其他系统程序一样。

1. 以 root 用户身份登录服务器计算机。
2. 运行 systemctl 命令，指定操作和实例名称：

```
systemctl start|stop|restart pki-tomcatd@instance_name.service
```

例如：

```
systemctl restart pki-tomcatd@pki-tomcat.service
```

13.2.2. 在机器重启后重启 PKI 实例

如果运行一个或多个 PKI 实例的计算机意外关机，则必须以正确顺序重启 PKI 实例多的服务，以便通过 HTML 服务页面和管理控制台提供子系统。

1. 如果子系统使用的 Directory 服务器实例安装在本地计算机上，请重新启动 Administration Server 和 Directory Server 进程。

```
systemctl start dirsrv-admin.service
systemctl start dirsrv@instance_name.service
```

2. 启动证书证书系统nbsp; 系统子系统实例。

```
systemctl start pki-tomcatd@instance_name.service
```

13.2.3. 检查 PKI 实例状态

systemctl 命令可用于检查进程的状态，显示进程是否正在运行或停止。例如：

```
systemctl -l status pki-tomcatd@pki-tomcat.service
pki-tomcatd@pki-tomcat.service - PKI Tomcat Server pki-tomcat
   Loaded: loaded (/lib/systemd/system/pki-tomcatd@.service; enabled)
   Active: inactive (dead) since Fri 2015-11-20 19:04:11 MST; 12s ago
  Process: 8728 ExecStop=/usr/libexec/tomcat/server stop (code=exited, status=0/SUCCESS)
  Process: 8465 ExecStart=/usr/libexec/tomcat/server start (code=exited, status=143)
  Process: 8316 ExecStartPre=/usr/bin/pkidaemon start tomcat %i (code=exited, status=0/SUCCESS)
 Main PID: 8465 (code=exited, status=143)

Nov 20 19:04:10 pki.example.com server[8728]: options used: -Dcatalina.base=/var/lib/pki/pki-tomcat
-Dcatalina.home=/usr/share/tomcat -Djava.endorsed.dirs= -Djava.io.tmpdir=/var/lib/pki/pki-
tomcat/temp -Djava.util.logging.config.file=/var/lib/pki/pki-tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
Nov 20 19:04:10 pki.example.com server[8728]: arguments used: stop
Nov 20 19:04:11 pki.example.com server[8465]: Nov 20, 2015 7:04:11 PM
org.apache.catalina.core.StandardServer await
Nov 20 19:04:11 pki.example.com server[8465]: INFO: A valid shutdown command was received via
the shutdown port. Stopping the Server instance.
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[before_stop]
```

```
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[stop]
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[configure_stop]
Nov 20 19:04:11 pki.example.com server[8465]: Nov 20, 2015 7:04:11 PM
org.apache.coyote.AbstractProtocol pause
Nov 20 19:04:11 pki.example.com server[8465]: INFO: Pausing ProtocolHandler ["http-bio-8080"]
Nov 20 19:04:11 pki.example.com systemd[1]: Stopped PKI Tomcat Server pki-tomcat.
```

如果实例正在运行，其状态检查会返回类似以下示例的信息：

```
systemctl -l status pki-tomcatd@pki-tomcat.service
pki-tomcatd@pki-tomcat.service - PKI Tomcat Server pki-tomcat
   Loaded: loaded (/lib/systemd/system/pki-tomcatd@.service; enabled)
   Active: active (running) since Fri 2015-11-20 19:09:09 MST; 3s ago
  Process: 8728 ExecStop=/usr/libexec/tomcat/server stop (code=exited, status=0/SUCCESS)
  Process: 9154 ExecStartPre=/usr/bin/pkidaemon start tomcat %i (code=exited, status=0/SUCCESS)
 Main PID: 9293 (java)
   CGroup: /system.slice/system-pki\x2dtomcatd.slice/pki-tomcatd@pki-tomcat.service
           ??????9293 java -DRESTEASY_LIB=/usr/share/java/resteasy-base -
Djava.library.path=/usr/lib64/nuxwdog-jni -classpath
/usr/share/tomcat/bin/bootstrap.jar:/usr/share/tomcat/bin/tomcat-juli.jar:/usr/share/java/commons-
daemon.jar -Dcatalina.base=/var/lib/pki/pki-tomcat -Dcatalina.home=/usr/share/tomcat -
Djava.endorsed.dirs= -Djava.io.tmpdir=/var/lib/pki/pki-tomcat/temp -
Djava.util.logging.config.file=/var/lib/pki/pki-tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.security.manager -
Djava.security.policy==/var/lib/pki/pki-tomcat/conf/catalina.policy
org.apache.catalina.startup.Bootstrap start

Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.core.StandardService startInternal
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Starting service Catalina
Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.core.StandardEngine startInternal
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Starting Servlet Engine: Apache
Tomcat/7.0.54
Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Deploying configuration descriptor /etc/pki/pki-
tomcat/Catalina/localhost/ROOT.xml
Nov 20 19:09:12 pki.example.com server[9293]: Nov 20, 2015 7:09:12 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:12 pki.example.com server[9293]: INFO: Deployment of configuration descriptor
/etc/pki/pki-tomcat/Catalina/localhost/ROOT.xml has finished in 2,071 ms
Nov 20 19:09:12 pki.example.com server[9293]: Nov 20, 2015 7:09:12 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:12 pki.example.com server[9293]: INFO: Deploying configuration descriptor /etc/pki/pki-
tomcat/Catalina/localhost/pki#admin.xml
```

13.2.4. 配置 PKI 实例以自动启动重启

systemctl 命令可用于在重启后自动启动实例。例如，以下命令会在重启后自动启动红帽管理服务

器、目录服务器和 CA：

```
# systemctl enable dirsrv-admin.service
# systemctl enable dirsrv.target
# systemctl enable pki-tomcatd@pki-tomcat.service
```

注意

默认的 PKI 实例安装和配置使用 pkispawn 命令，让实例在重新引导时启动。

要禁用此行为（即，为了防止 PKI 实例在重新引导时自动启动），请发出下列命令：

```
# systemctl disable pki-tomcatd@pki-tomcat.service
# systemctl disable dirsrv.target
# systemctl disable dirsrv-admin.service
```

13.2.5. 为证书证书 Systemnbsp 设置 sudo 权限; 系统服务

为了简化管理和安全性，可以配置证书证书 Systemnbsp;System 和 Directory Server 进程，以便配置 PKI 管理员（而不是仅 root 用户）可以启动和停止服务。

设置子系统时的建议选项是使用 pkiadmin 系统组。（详情请参考 Red Hat Certificate System 9 规划、安装和部署指南。）所有为 CertificateCertificate Systemnbsp 的操作系统用户; 然后，系统管理员会添加到该组中。如果存在 pkiadmin 系统组，则可以被授予 sudo 访问权限来执行某些任务。

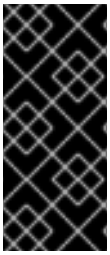
1. *编辑 /etc/sudoers 文件；在 Red Hat Enterprise Linuxnbsp;Hat Enterprise Linuxnbsp;Linux 7 中，可以使用 visudo 命令实现这一目的：*

   ```
   # visudo
   ```

2. *根据机器上安装的内容，为 Directory Server、管理服务器、PKI 管理工具和每个 PKI 子系统实例添加一行，为 pkiadmin 组授予 sudo 权限：*

   ```
   # For Directory Server services
   %pkiadmin ALL = PASSWD: /usr/bin/systemctl * dirsrv.target
   %pkiadmin ALL = PASSWD: /usr/bin/systemctl * dirsrv-admin.service

   # For PKI instance management
   %pkiadmin ALL = PASSWD: /usr/sbin/pkispawn *
   %pkiadmin ALL = PASSWD: /usr/sbin/pkidestroy *
   ```

```
# For PKI instance services
%pkiadmin ALL = PASSWD: /usr/bin/systemctl * pki-tomcatd@instance_name.service
```



重要

确保为每台证书证书系统nbsp;System、Directory Server 和 Administration Server 在机器上设置 sudo 权限 - 以及 仅适用于 机器上的那些实例。计算机上可能存在同一子系统类型的多个实例，或者没有子系统类型的实例。它取决于部署。

13.3. 打开 SUBSYSTEM 控制台和服务

每个子系统具有不同的接口，可供不同的用户类型访问。除 TKS 外，所有子系统都有某种 Web 服务页面，适用于代理、管理员或最终用户（或全部三个）。此外，CA、KRA、OCSP 和 TKS 均具有基于 Java 的控制台，必须在服务器上安装该控制台，以执行管理任务来管理子系统本身。

对于基于 Web 的 Web 服务页面而言，可以定制基于 Web 的外观和有限程度的功能，以更好地与组织的现有网站集成。请参阅 红帽认证系统规划、安装和部署指南。

13.3.1. 查找 subsystem Web Services 页面

CA、KRA、OCSP、TKS 和 TPS 子系统有用于代理、常规用户和管理员的 Web 服务页面。可通过通过子系统的安全最终用户端口打开到子系统主机的 URL 来访问这些 Web 服务菜单。例如，对于 CA：

```
https://server.example.com:8443/ca/services
```

每个子系统的主要 Web 服务页面都有可用的服务页面列表，在 表 13.1 “默认网页” 中进行了概述。要具体访问任何服务，请访问适当的端口，并将适当的目录附加到 URL。例如，访问 CA 的最终实体（普通用户）Web 服务：

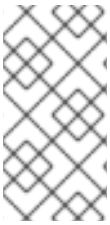
```
https://server.example.com:8443/ca/ee/ca
```

如果正确配置了 DNS，那么可以使用 IPv4 或 IPv6 地址连接到服务页面。例如：

```
https://1.2.3.4:8443/ca/services
https://[00:00:00:00:123:456:789:00:]:8443/ca/services
```

有些子系统接口需要客户端身份验证才能访问，通常是与代理或管理员角色关联的接口。其他接口，即使在安全（SSL 连接）中运行也不需要客户端身份验证。其中一些接口（如端到端的实体服务）可以配

置为要求进行客户端身份验证，但其他部分不能被配置为支持客户端身份验证。这些区别包括在 表 13.1 “默认网页” 中。



注意

任何人都可以访问子系统的最终用户页面，但访问代理或管理 Web 服务页面都需要在 Web 浏览器中发布并安装代理或管理员证书，或者对 Web 服务进行身份验证会失败。

表 13.1. 默认网页

用于 SSL	用于客户端身份验证[a]	Web 服务	Web 服务位置
证书管理器			
否		结束实体	ca/ee/ca/
是	否	结束实体	ca/ee/ca
是	是	代理	ca/agent/ca
是	否	服务	ca/services
是	否	控制台（Console）	pkiconsole https://*host:port*/ca
密钥恢复授权			
是	是	代理	kra/agent/kra
是	否	服务	kra/services
是	否	控制台（Console）	pkiconsole https://*host:port*/kra
在线证书状态管理器			
是	是	代理	ocsp/agent/ocsp
是	否	服务	ocsp/services
是	否	控制台（Console）	pkiconsole https://*host:port*/ocsp
令牌密钥服务			
是	否	服务	tks/services

用于 SSL	用于客户端身份验证[a]	Web 服务	Web 服务位置
是	否	控制台（Console）	pkiconsole https://*host:port*/tks
令牌处理系统			
是		服务	index.cgi

[a] 带有客户端身份验证值的服务可以重新配置为 **No**，以要求进行客户端身份验证。没有 **Yes** 或 **No** 值的服务不能配置为使用客户端身份验证。

13.3.2. 启动证书证书 Systemnbsp;System Administrative Console

控制台是通过使用 pkiconsole 命令通过其 SSL 端口连接到子系统实例而打开的。这个命令的格式如下：

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

subsystem_type 可以是 ca、kra、ocsp 或 tks。例如，这会打开 KRA 控制台：

```
pkiconsole https://server.example.com:8443/kra
```

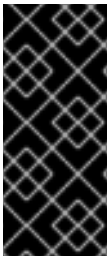
如果正确配置了 DNS，那么可以使用 IPv4 或 IPv6 地址连接到控制台。例如：

```
pkiconsole https://1.2.3.4:8443/ca
pkiconsole https://[00:00:00:00:123:456:789:00:]:8443/ca
```

13.3.3. 为 Java 管理控制台启用 SSL

对证书证书证书系统nbsp 的基于证书的验证可以启用系统控制台，以便管理员必须先使用客户端证书进行身份验证，然后才能登录到证书证书系统nbsp;System Console.在启用基于证书的身份验证前，存储管理员的证书。

要在控制台中启用 SSL，请同时配置客户端和服务器。



重要

如果 CA 配置了通过管理端口进行客户端身份验证，并且该 CA 是安全域管理器，则无法配置将 CA 用于其安全域 的新 PKI 子系统。新的 PKI 实例通过 admin 端口注册到安全域 CA，但不使用客户端身份验证。如果 CA 需要客户端身份验证，则注册尝试会失败。

首先，设置证书Certificate Systemnbsp;System server 以使用 SSL 客户端验证：

1. **使用这个系统的任何管理员存储证书。证书应该来自 CA 本身，或者从哪个 CA 签署该子系统的证书。**

   a. **打开子系统控制台。**

   b. **选择左侧的 "用户和组" 选项。**

   c. **在 Users 选项卡中，选择管理用户，再单击 Manage Certificates。**

   d. **点 Import。**

   e. **粘贴到 base-64 编码的 SSL 客户端证书中，如保存在 web 浏览器中的管理员证书。**

   确保客户端证书适用于 SSL 客户端身份验证；否则，服务器将不接受客户端证书，并将在 /var/log/instanceID/system 中的错误日志中发布错误消息：

   ```
   failure (14290): Error receiving connection
   SEC_ERROR_INADEQUATE_CERT_TYPE - Certificate type not approved for application.)
   ```

2. **停止子系统。**

   ```
   systemctl stop pki-tomcatd@instance_name.service
   ```

3. **打开实例配置目录 /var/lib/pki/instance_name/subsystem_type/conf。**

4. 打开文件 CS.cfg。

5. 将 authType 参数的值从 pwd 更改为 sslclientauth ：

```
authType=sslclientauth
```

6. 保存该文件。

7. 打开 server.xml 文件。

8. 将 admin 接口连接器部分中的 clientAuth="false" 属性更改为 clientAuth="want "。

```
<Connector port="8443" maxHttpHeaderSize="8192"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" disableUploadTimeout="true"
        acceptCount="100" scheme="https" secure="true"
        clientAuth="want" sslProtocol="SSL"
.....
        serverCertFile="/var/lib/pki/pki-tomcat/conf/serverCertNick.conf"
        passwordFile="/var/lib/pki/pki-tomcat/conf/password.conf"
        passwordClass="org.apache.tomcat.util.net.jss.PlainPasswordFile"
        certdbDir="/var/lib/pki/pki-tomcat/alias"/>
```

want 值表示首选客户端身份验证，但不是必需的。这样，客户端通过可轻松使用它（如控制台）的接口进行身份验证，同时仍然允许客户端验证（安全域内的其他子系统）使用常规连接进行连接。

9. 启动子系统。

```
systemctl start pki-tomcatd@instance_name.service
```

设置服务器后，将客户端配置为使用 SSL 客户端身份验证。

控制台必须具有对服务器 SSL 客户端身份验证的管理员证书和密钥的访问权限。控制台的默认证书和密钥数据库存储在 .redhat-idm-console 目录中。

要提供对管理员证书和密钥的访问，请从管理员的浏览器中将它们导出到 .p12 文件中，然后使用 pk12util 导入该文件，或者将浏览器的证书和密钥数据库复制到 .redhat-idm-console 目录中。（这个步骤假定证书从浏览器导出到 .p12 文件中。）

1. *将管理员用户证书和密钥从浏览器导出到文件，如 admin.p12。*

2. *打开用户的控制台目录。*

```
/user-directory/.redhat-idm-console
```

3. *如有必要，创建新的安全数据库。*

```
certutil -N -d .
```

4. *停止证书证书 Systemnbsp;System 实例。*

```
systemctl stop pki-tomcatd@instance_name.service
```

5. *使用 pk12util 导入证书。*

```
# pk12util -i /tmp/admin.p12 -d /user-directory/.redhat-idm-console -W [p12filepassword]
```

如果这个过程成功，命令会输出以下内容：

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

6. *从浏览器中导出发布 CA 证书的 64 位 Blob，并将其保存到类似 ca.crt 的文件。*

7. *从与 admin 用户证书关联的基本 64blob 中导入 CA 证书。*

```
certutil -A -d . -n ca -t CT,C,C -i ./ca.crt
```

8. *启动证书证书系统nbsp;System 实例。*

```
systemctl start pki-tomcatd@instance_name.service
```

9. 启动控制台；现在，它会提示输入证书。

13.4. 在 JAVA 安全管理器下运行子系统

Java 服务具有安全管理器的选项，它为应用程序定义了不安全的安全操作。安装子系统后，它们会自动启用安全管理器，即每个 Tomcat 实例从运行安全管理器开始。

13.4.1. 关于安全管理器策略文件

当五个 Java 子系统（CA、OCSP、KRA、TKS 和 TPS）在 Java Security Manager 中运行时，它们使用了两组策略的组合：

- 来自 /usr/share/tomcat/conf 目录中的默认 Tomcat 策略中的 catalina.policy 文件；每当更新常规 Tomcat 文件时更新。
- 在 /var/lib/pki/instance_name/subsystem_type/conf 目录中提供的 pki.policy 文件，它随子系统实例提供。
- 在 /var/lib/pki/instance_name/subsystem_type/conf 目录中有一个 custom.policy 文件，其中包含用户定义的安全策略。

每当 Tomcat 服务开始创建修订的 catalina.policy 文件时，这三个文件也会连接在一起，该文件也位于 /var/lib/pki/instance_name/subsystem_type/conf 目录中。

默认的 pki.policy 文件包含授予 PKI 子系统使用的 Tomcat、LDAP 和 symkey 服务对不受限制的访问权限的权限。例如：

```
// These permissions apply to Tomcat java as utilized by PKI instances
grant codeBase "file:/usr/share/java/tomcat/-" {
    permission java.security.AllPermission;
};
```

custom.policy 文件默认为空；管理员可以在文件中写入策略，除了给定的 PKI 策略和 Tomcat 策略外，还会使用该文件。

13.4.2. 在不使用 Java 安全管理器的情况下启动 subsystem 实例

在 Java 安全管理器下，在 PKI Tomcat 实例中配置的所有 Java 子系统自动运行（除非通过覆盖 /etc/pki/default.cfg 文件的 [Tomcat] 部分下的 pki_security_manager=true 创建）但是，可以启动或重启实例，并在不启动 Java 安全管理器的情况下运行 它，如下所示。

过程 13.1. 在没有 Java Security Manager 的情况下启动实例

1. 停止该实例。

```
# systemctl stop pki-tomcatd@instance_name.service
```

2. 编辑 /etc/sysconfig/instance_name 文件并关闭安全管理器：

```
SECURITY_MANAGER="false"
```

3. 启动实例。

```
# systemctl start pki-tomcatd@instance_name.service
```

13.5. 配置 LDAP 数据库

CertificateCertificate Systemnbsp;System 执行证书和密钥管理功能以响应它接收的请求。这些功能包括：

- 存储和检索证书请求
- 存储和检索证书记录
- 存储 CRL
- 存储 ACL
- 存储特权用户和角色信息

- 存储和检索最终用户的加密密钥记录

为实现这些功能，证书证书证书系统：System 与 Red Hat Directory Server（称为内部数据库或 本地数据库 ）整合在一起。目录服务器被引用为 CertificateCertificate Systemnbsp;System configuration;System configuration; when CertificateCertificate Systemnbsp;System 子系统时，会在 Directory Server 中创建一个新数据库。这个数据库由 CertificateCertificate Systemnbsp 专门用作嵌入式数据库；System instance and 可以使用 Directory Server 附带的目录管理工具进行管理。

CertificateCertificate Systemnbsp;System instance 数据库列出了 serverRoot/slapd-DS_name/db/ 目录中的其他 Directory Server 数据库。这些数据库通过 /etc/pki/default.cfg 文件中的 pki_ds_database 变量的值确定的值（CS_instance_name-CA,CS_instance_name-KRA,CS_instance_name-OCSP OCSP ,CS_instance_name-TKS, CS_instance_name-TPS 默认是实例配置期间给出的默认格式。例如，对于名为 ca1 的证书管理器，数据库名称为 ca1-CA。同样，数据库名称由 /etc/pki/default.cfg 文件中的 pki_ds_base_dn 变量的值决定，（o=CS_instance_name-CA, o=CS_instance_name-KRA, o=CS_instance_name-OCSP, o=CS_instance_name-OCSP, O=CS_instance_name-TKS，默认是 o=CS_instance_name-TPS，也在配置期间设置。

子系统使用数据库来存储不同的对象。证书管理器存储所有数据、证书请求、证书、CRL 和相关信息，而 KRA 则只存储密钥记录和相关数据。

警告

内部数据库架构配置为仅存储证书证书系统nbsp；系统数据。不要对其进行任何更改，或者配置证书证书系统nbsp;System 以使用任何其他 LDAP 目录。这样做可能会导致数据丢失。

另外，请勿将内部 LDAP 数据库用于任何其他目的。

13.5.1. 更改内部数据库配置

要更改子系统实例用作内部数据库的 Directory 服务器实例：

1. 登录子系统管理控制台。

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

2. 在 Configuration 选项卡中，选择 Internal Database 选项卡。

3. 通过更改主机名、端口和绑定 DN 字段来更改目录服务器实例。

   hostname 是安装 Directory 服务器的机器的完全限定主机名，如 certificate.example.com。CertificateCertificate Systemnbsp;System 使用此名称来访问目录。

   默认情况下，用作内部数据库的 Directory 服务器实例的主机名显示为 localhost，而不是实际主机名。这是为了防止内部数据库在系统外部可见，因为 localhost 上的服务器只能从本地计算机进行访问。因此，默认配置可最大程度减少从本地机器外连接到这个目录服务器实例的风险。

   如果内部数据库的可见性仅限于本地子网，可以将主机名更改为 localhost 以外的其他名称。例如，如果证书Certificate Systemnbsp;System 和 Directory Server 安装在单独的机器上以进行负载平衡，请指定安装目录服务器的机器的主机名。

   端口号是用于与 Directory 服务器的非 SSL 通信的 TCP/IP 端口。

   DN 应该是 Directory Manager DN。CertificateCertificate Systemnbsp;System 子系统在访问目录树与目录通信时使用此 DN。

4. 点 Save。

   配置已被修改。如果更改需要重新启动服务器，则会出现包含该消息的提示。在这种情况下，重启服务器。

13.5.2. 在目录服务器中使用证书系统发布的证书

要在安装证书系统时使用加密连接，需要使用外部证书颁发机构(CA)或自签名证书签发的证书。但是，在设置证书系统 CA 后，管理员通常希望用证书系统发布的证书替换此证书。

将目录服务器使用的 TLS 证书替换为证书系统发布的证书：

1. **在 Directory Server 主机上：**

   a. **停止 Directory 服务器实例：**

   ```
   # systemctl stop dirsrv@instance_name
   ```

   b. **生成证书签名请求(CSR)。**

   例如，生成一个使用 2048 位 RSA 加密的 CSR，并将其存储在 ~/ds.csr 文件中：

   ```
   # PKCS10Client -d /etc/dirsrv/slapd-instance_name/ -p password -a rsa -l 2048 -o
   ~/ds.csr -n "CN=$HOSTNAME"
   PKCS10Client: Debug: got token.
   PKCS10Client: Debug: thread token set.
   PKCS10Client: token Internal Key Storage Token logged in...
   PKCS10Client: key pair generated.
   PKCS10Client: CertificationRequest created.
   PKCS10Client: b64encode completes.
   Keypair private key id: -3387b397ebe254b91c5d6c06dc36618d2ea8b7e6

   -----BEGIN CERTIFICATE REQUEST-----
   ...
   -----END CERTIFICATE REQUEST-----
   PKCS10Client: done. Request written to file: ~/ds.csr
   ```

   c. **启动 Directory 服务器实例，以启用 CA 来处理请求：**

   ```
   # systemctl start dirsrv@instance_name
   ```

   d. **将 CSR 提交到证书系统的 CA。例如：**

   ```
   # pki -d /etc/dirsrv/slapd-instance_name/ ca-cert-request-submit --profile caServerCert --
   csr-file ~/ds.csr
   ----------------------------
   Submitted certificate request
   ----------------------------
     Request ID: 13
     Type: enrollment
     Request Status: pending
     Operation Result: success
   ```

2.

2.

在证书系统主机上：

a.

将 CA 代理证书导入到网络安全服务(NSS)数据库中以签署 CMC 完整请求：

i.

创建新目录。例如：

```
# mkdir ~/certs_db/
```

ii.

在新创建的目录中初始化数据库：

```
# certutil -N -d ~/certs_db/
```

iii.

显示 CA 签名证书的序列号：

```
# pki -p 8080 ca-cert-find --name "CA Signing Certificate"
---------------
1 entries found
---------------
Serial Number: 0x87bbe2d
...
```

iv.

使用上一步中的序列号，将 CA 签名证书下载到 ~/certs_db/CA.pem 文件中：

```
# pki -p 8080 ca-cert-show 0x87bbe2d --output ~/certs_db/CA.pem
```

v.

将 CA 签名证书导入到 NSS 数据库中：

```
# pki -d ~/certs_db/ -c password client-cert-import "CA Certificate" --ca-cert
~/certs_db/CA.pem
```

vi.

导入代理证书：

```
# pk12util -d ~/certs_db/ -i ~/.dogtag/instance_name/ca_admin_cert.p12
Enter Password or Pin for "NSS FIPS 140-2 Certificate DB": password
Enter password for PKCS12 file: password
pk12util: PKCS12 IMPORT SUCCESSFUL
```

b.

b.

通过 CMS(CMC)请求创建证书管理：

i.

创建一个配置文件，如 ~/sslserver-cmc-request.cfg，其中包含以下内容：

```
# NSS database directory where the CA agent certificate is stored.
dbdir=~/certs_db/

# NSS database password.
password=password

# Token name (default is internal).
tokenname=internal

# Nickname for CA agent certificate.
nickname=caadmin

# Request format: pkcs10 or crmf.
format=pkcs10

# Total number of PKCS10/CRMF requests.
numRequests=1

# Path to the PKCS10/CRMF request.
# The content must be in Base-64 encoded format.
# Multiple files are supported. They must be separated by space.
input=~/ds.csr

# Path for the CMC request.
output=~/sslserver-cmc-request.bin
```

ii.

创建 CMC 请求：

```
# CMCRequest ~/sslserver-cmc-request.cfg
...
The CMC enrollment request in base-64 encoded format:
...
The CMC enrollment request in binary format is stored in ~/sslserver-cmc-
request.bin
```

c.

提交 CMC 请求：

i.

创建一个配置文件，如 ~/sslserver-cmc-submit.cfg，其中包含以下内容：

```
# PKI server host name.
host=server.example.com
```

```
# PKI server port number.
port=8443

# Use secure connection.
secure=true

# Use client authentication.
clientmode=true

# NSS database directory where the CA agent certificate is stored.
dbdir=~/certs_db/

# NSS database password.
password=password

# Token name (default: internal).
tokenname=internal

# Nickname of CA agent certificate.
nickname=caadmin

# CMC servlet path
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCserverCert

# Path for the CMC request.
input=~/sslserver-cmc-request.bin

# Path for the CMC response.
output=~/sslserver-cmc-response.bin
```

ii. ***提交请求：***

```
# HttpClient sslserver-cmc-submit.cfg
...
The response in binary format is stored in
~/sslserver-cmc-response.bin
```

iii. ***（可选）验证结果：***

```
# CMCResponse -d ~/certs_db/ -i ~/sslserver-cmc-response.bin
...
Number of controls is 1
Control #0: CMCStatusInfoV2
  OID: {1 3 6 1 5 5 7 7 25}
  BodyList: 1
  Status: SUCCESS
```

d. ***显示 Directory 服务器证书的序列号：***

```
# pki -p 8080 ca-cert-find --name "DS Certificate"
---------------
1 entries found
---------------
Serial Number: 0xc3eeb0c
...
```

e. **使用上一步中的序列号下载证书：**

```
# pki -p 8080 ca-cert-show 0xc3eeb0c --output ~/ds.crt
```

f. **将 Directory Server 和 CA 证书的证书复制到 Directory Server 主机上。例如：**

```
# scp ~/ds.crt ~/certs_db/CA.pem ds.example.com:~/
```

g. **停止证书系统：**

```
# systemctl stop pki-tomcatd@instance_name.service
```

3. **在 Directory Server 主机上：**

a. **停止 Directory 服务器实例：**

```
# systemctl stop dirsrv@instance_name
```

b. **替换证书。详情请查看 Red Hat Directory Server Administration Guide 中的对应部分：**

i. **删除旧的证书和密钥 CA 证书。请参阅 删除证书。**

ii. **安装证书系统发布的 CA 证书。请参阅安装 CA 证书。**

iii. **为证书系统发布的目录服务器安装证书。请参阅安装证书。**

c.

c.

启动 Directory 服务器实例：

```
# systemctl start dirsrv@instance_name
```

4.

启动证书系统：

```
# systemctl stop pki-tomcatd@instance_name.service
```

5.

另外，还可配置基于证书的身份验证。详情请查看 第 13.5.3 节 “使用内部数据库启用 SSL/TLS 客户端身份验证”。

13.5.3. 使用内部数据库启用 SSL/TLS 客户端身份验证

客户端身份验证 允许一个实体通过提供证书向另一个实体进行身份验证。此验证方法由证书证书系统 nbsp 使用；例如， System 代理登录到代理服务页面。

要在证书证书系统nbsp 之间使用 SSL/TLS 连接；系统实例和 LDAP 目录服务器使用其内部数据库，必须启用客户端身份验证以允许证书证书证书认证系统nbsp；System 实例可以进行身份验证并绑定到 LDAP 目录。

设置客户端身份验证有两个部分。第一种方法是配置 LDAP 目录，如设置 SSL/TLS 并将 ACI 设置为控制证书Certificate Systemnbsp;System 实例访问。第二个是在 CertificateCertificate Systemnbsp;System 实例创建用户， 它将用来绑定到 LDAP 目录并设置其证书。

要为 PKI 实例配置 LDAPS， 请参阅 pkispawn(8) man page（示例：使用安全 LDAP 连接安装 PKI 子系统）。

13.5.4. 限制对内部数据库的访问

Red Hat Directory Server Console 显示证书Certificate Systemnbsp 的 Directory Server 实例的条目或图标；系统使用 作为其内部数据库。

与证书证书Systemnbsp 不同的是：System Console 仅限于具有证书证书系统nbsp 的用户；系统管理员权限，目录服务器控制台可供任何用户访问。用户可以为内部数据库打开目录服务器控制台，并更改为存储的数据，例如从证书证书系统nbsp中删除用户；System administrator group 或将自己的条目添加到组中。

可将访问权限限制为仅限那些知道目录管理器 DN 和密码的用户。可以通过修改单点登录密码缓存来更改此密码。

1. 登录到 Directory 服务器控制台。

2. 选择 CertificateCertificate Systemnbsp;System internal database 条目，然后单击 Open。

3. 选择 Configuration 选项卡。

4. 在导航树中，展开 Plug-ins 并选择 Pass- via Authentication。

5. 在右窗格中，取消选择 启用插件 复选框。

6. 点 Save。

   服务器提示重新启动服务器。

7. 点 Tasks 选项卡，然后点 Restart the Directory Server。

8. 关闭 Directory 服务器控制台。

9. 服务器重启时，打开内部数据库实例的 Directory 服务器控制台。

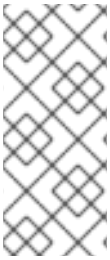
   此时会出现 Login to Directory 对话框; Distinguished Name 字段显示 Directory Manager DN; 输入密码。

   只有在输入了正确的密码时，内部数据库的 Directory 服务器控制台才会打开。

13.6. 查看安全域配置

安全域 是 PKI 服务的注册表。PKI 服务（如 CA）在这些域中注册有关自身的信息，以便 PKI 服务用户可以通过检查注册表来查找其他服务。CertificateCertificate Systemnbsp 中的安全域服务;System 为证书证书系统nbsp;System 子系统和一组共享的信任策略同时管理 PKI 服务注册。

安全域自动管理子系统之间的信任关系，因此如果 TPS、TKS 和 KRA 位于相同的安全域中，它们可以安全地通信。



注意

安全域在子系统配置中使用。设置子系统后，它可以检查安全域 registry，以查看可用的实例。如果需要与另一个实例（如使用 TKS 和 KRA）操作的 TPS 和 KRAS 创建可信关系，则会使用安全域在所选 TKS 和 KRA 实例上创建 TPS 代理用户。

注册表提供对该域中子系统提供的所有 PKI 服务的完整视图。每个证书证书系统nbsp;System 子系统必须是主机或者安全域的成员。

只有 CA 可以托管和管理安全域。每个 CA 都有自己的 LDAP 条目，而安全域是该 CA 条目下面的机构组：

```
ou=Security Domain,dc=example,dc=com
```

然后，在安全域组织组下有一个每个子系统类型的列表，具有特殊对象类(pkiSecurityGroup)来识别组类型：

```
cn=KRAList,ou=Security Domain,dc=example,dc=com
objectClass: top
objectClass: pkiSecurityGroup
cn: KRAList
```

然后，每个子系统实例都作为该组的成员存储，并带有一个特殊的 pkiSubsystem对象类来标识条目类型：

```
dn: cn=server.example.com:8443,cn=KRAList,ou=Security Domain,dc=example,dc=com
objectClass: top
objectClass: pkiSubsystem
cn: kra.example.com:8443
host: server.example.com
SecurePort: 8443
SecureAgentPort: 8443
SecureAdminPort: 8443
```

> UnSecurePort: 8080
> DomainManager: false
> Clone: false
> SubsystemName: KRA server.example.com 8443

13.7. 管理 子系统的 SELINUX 策略

SELinux 是强制访问控制规则的集合，用于限制未经授权的访问和篡改。有关 SELinux 的详情，请查看 SELinux 用户和管理员指南。

13.7.1. 关于 SELinux

基本上，SELinux 识别系统上的 对象，可以是文件、目录、用户、进程、套接字或 Linux 主机上的任何其他操作。这些对象对应于 Linux API 对象。然后，每个对象 都映射到安全上下文，它定义了对象的类型以及如何在 Linux 服务器上工作。

系统进程在 SELinux 域中运行。每个域都有一组规则，用于定义 SELinux 域如何与系统中的其他 SELinux 对象交互。然后，这组规则决定进程可以访问哪些资源以及它可以对这些资源执行的操作。

对于 CertificateCertificate Systemnbsp;System，每个子系统类型都在该子系统类型的特定域中运行。该子系统类型的每个实例都属于同一个 SELinux 域，无论系统中有多少个实例，例如，如果服务器上安装了三个 CA，则所有三个实例都属于 http_port_t SELinux 域。

所有子系统的规则和定义组成了整个证书证书系统空间；系统 SELinux 策略。CertificateCertificate Systemnbsp; 安装子系统时，系统 SELinux 策略已经配置，并且每次使用 pkispawn 添加子系统时，都会更新所有 SELinux 策略，或使用 pkidestroy 删除。

CertificateCertificate Systemnbsp;System 子系统使用 SELinux 设置运行，这意味着即使需要所有 SELinux 规则，也可以成功执行证书证书系统操作。

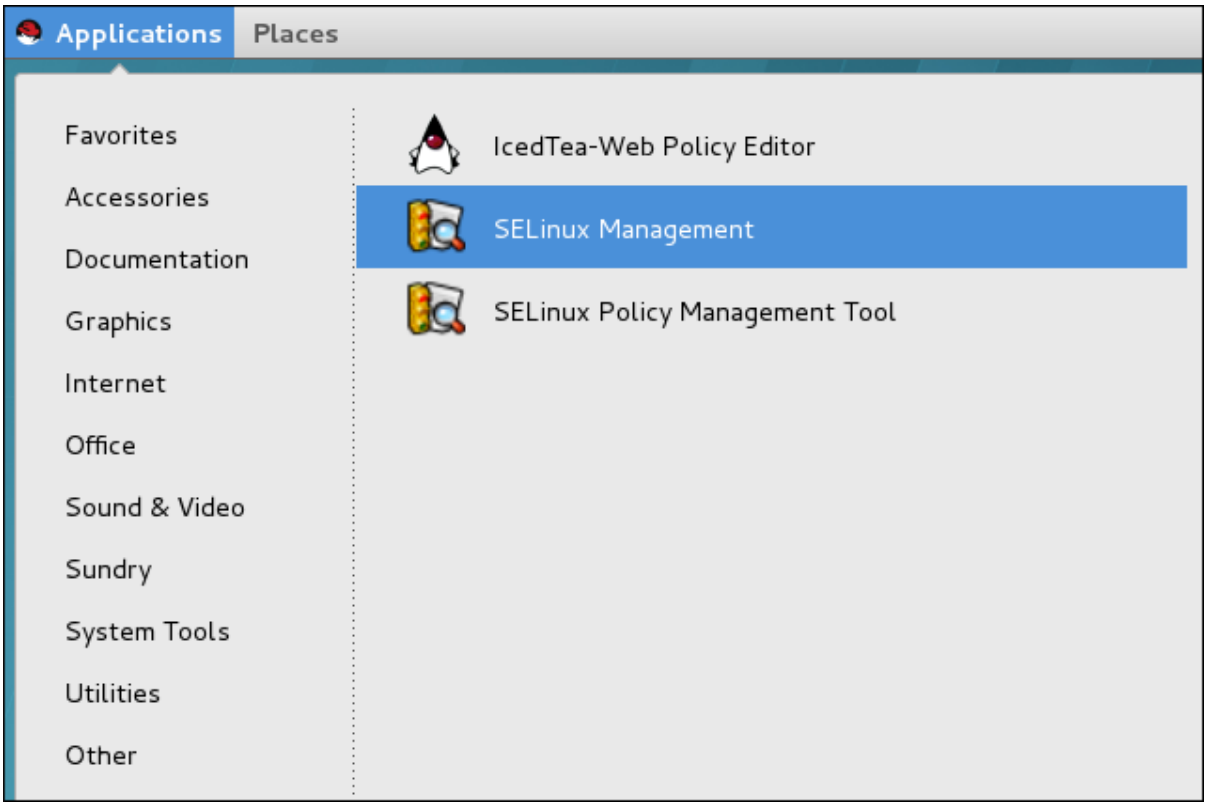
默认情况下，证书Certificate Systemnbsp;System 子系统受 SELinux 策略限制。

13.7.2. 查看子系统的 SELinux 策略

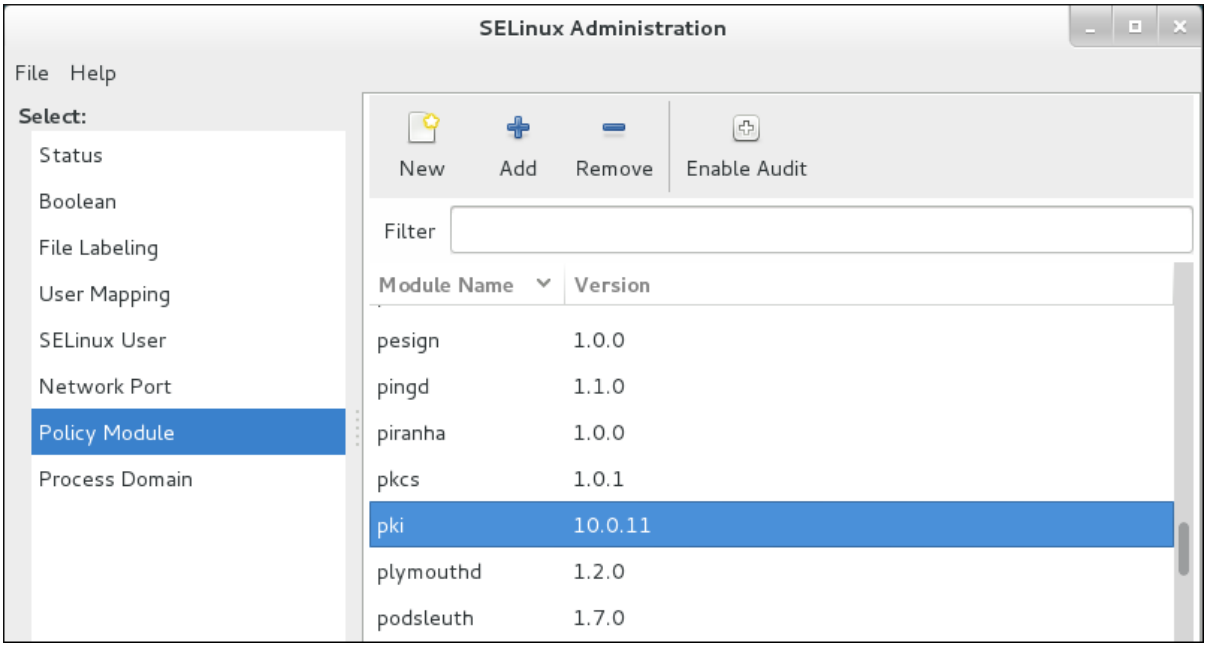
所有证书证书系统启动;系统策略都是系统 SELinux 策略的一部分。可以使用 SELinux 管理 GUI 查看配置的策略，您可以通过安装 policycoreutils-gui 软件包来获取该策略。

1. 运行 system-config-selinux 命令，或通过访问主系统菜单的 Applications → Other →

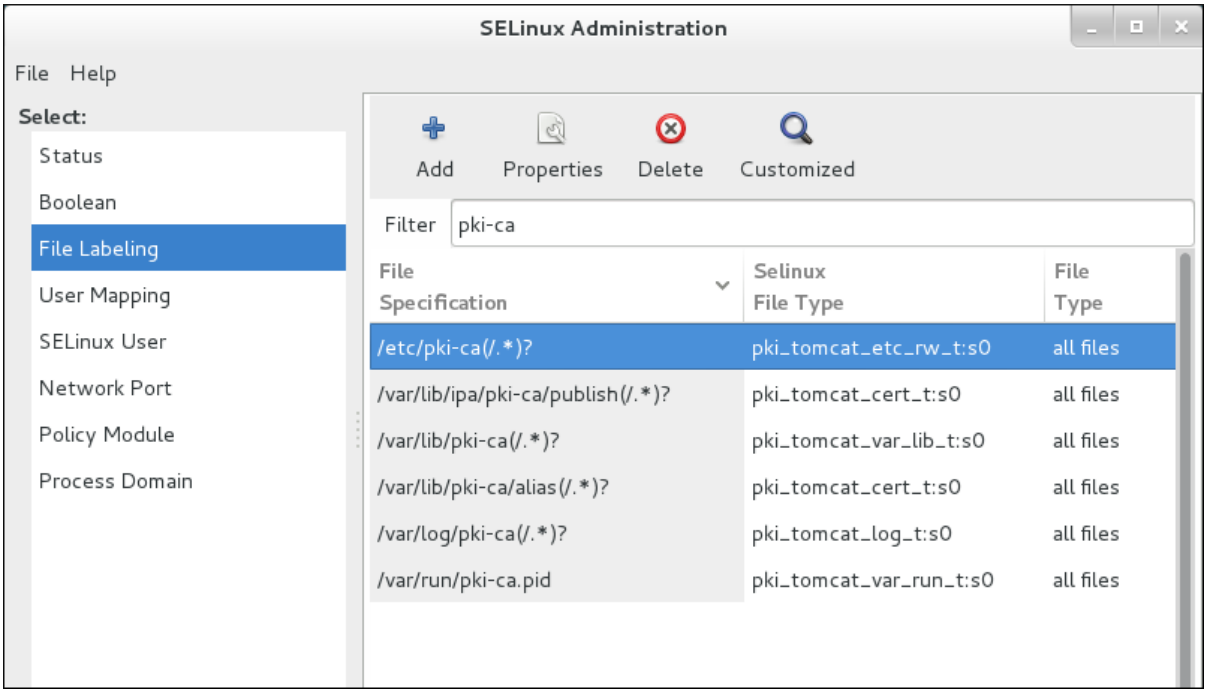
SELinux Management 来打开该实用程序。



2. *要检查 CertificateCertificate Systemnbsp;System SELinux 策略的版本，请单击左侧栏中的 Policy Module 部分。*



3. *要查看单个文件和进程上设置的策略，请点击 File Labeling 部分。若要查看子系统的端口分配的策略，请单击 Network Port 部分。*



13.7.3. 重新标记 nCipher netHSM 上下文

nCipher netHSM 软件不附带自己的 SELinux 策略，因此 CertificateCertificate Systemnbsp;System 包含了一个默认的 netHSM 策略，如 例 13.1 “netHSM SELinux Policy”。

例 13.1. netHSM SELinux Policy

```
# default labeling for nCipher
/opt/nfast/scripts/init.d/(.*)  gen_context(system_u:object_r:initrc_exec_t,s0)
/opt/nfast/sbin/init.d-ncipher  gen_context(system_u:object_r:initrc_exec_t,s0)
/opt/nfast(/.*)?                gen_context(system_u:object_r:pki_common_t, s0)
/dev/nfast(/.*)?                gen_context(system_u:object_r:pki_common_dev_t,0)
```

其他规则允许 pki_*_t 域与标记为 pki_common_t 和 pki_common_dev_t 的文件进行通信。

如果 nCipher 配置被改变（即使它位于默认目录 /opt/nfast中），请运行 restorecon 以确保正确标记所有文件：

```
restorecon -R /dev/nfast
restorecon -R /opt/nfast
```

如果 nCipher 软件安装在不同的位置，或者使用了其他 HSM，则默认证书证书系统nbsp;System HSM 策略需要使用 semanage 重新标记。

13.8. 备份和恢复证书证书证书NBSP;系统

CertificateCertificate Systemnbsp;System 不包括备份和恢复工具。但是，证书证书系统证书系统仍然可以被手动归档和恢复，如果证书或密钥信息丢失，则仍可访问信息的部署。CertificateCertificate Systemnbsp 的三个主要部分；在数据丢失或硬件故障时，需要定期备份系统：

- 内部数据库。子系统使用 LDAP 数据库来存储其数据。目录服务器提供自己的备份脚本和程序。
- 安全数据库。安全数据库存储证书和密钥资料。如果这些都存储在 HSM 中，请查阅 HSM 供应商文档来获取如何备份数据的信息。如果信息存储在实例 别名 目录中的默认目录中，则会使用 instance 目录进行备份。要单独备份，请使用 tar 或 zip 等实用程序。
- 实例目录.instance 目录包含所有配置文件、安全数据库和其他实例文件。这可以使用 tar 或 zip 等实用程序备份。

13.8.1. 备份和恢复 LDAP 内部数据库

Red Hat Directory Server 文档 包含备份和恢复数据库的更多详细信息。

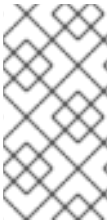
13.8.1.1. 备份 LDAP 内部数据库

可使用两个工具对备份 Directory 服务器实例；每个备份工具都有一个对应部分来恢复它生成的文件：

- db2ldif 工具会创建一个 LDIF 文件，您可以使用 ldif2db 工具恢复。
- db2bak 命令创建一个备份文件，您可以使用 bak2db 工具恢复。

13.8.1.1.1. 使用 db2ldif 备份

运行 db2ldif 命令可备份单个子系统数据库，如 -n 选项指定。



注意

当 db2ldif 命令使用 dirsrv 用户运行时，它不具有在 /root/ 目录下写入的权限，因此您需要提供可写入的路径。

1. 备份 PKI 子系统使用的每个目录服务器数据库。您可以使用 pki-server ca-db-config-show 命令检查给定子系统的数据库名称。

   例如：

```
# db2ldif -V -n pki-tomcat-CA -a /var/lib/dirsrv/slapd-pki1/ldif/pki-ca-backup.ldif
Exported ldif file: /var/lib/dirsrv/slapd-pki1/ldif/pki-ca-backup.ldif
ldiffile: /var/lib/dirsrv/slapd-pki1/ldif/pki-ca-backup.ldif
[05/Nov/2020:10:17:53.835635923 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[05/Nov/2020:10:17:53.845938266 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[05/Nov/2020:10:17:53.851851787 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[05/Nov/2020:10:17:53.874058831 -0500] - INFO - ldbm_back_ldbm2ldif - export pki-tomcat-CA: Processed 67 entries (100%).
[05/Nov/2020:10:17:53.884181122 -0500] - INFO - dblayer_pre_close - All database threads now stopped
```

2. 除了备份所有单独的子数据库外，您还可以通过添加 用户Root 作为 -n 选项备份主数据库。例如：

```
# db2ldif -V -n userRoot -a /var/lib/dirsrv/slapd-pki1/ldif/userRoot.ldif
```

要使用 ldif2db 恢复 LDIF 文件，请参阅 第 13.8.1.2.1 节 “使用 ldif2db 恢复”。

13.8.1.1.2. 使用 db2bak 备份

运行 db2bak 命令备份所有证书证书系统nbsp；该目录服务器的系统子系统数据库（以及由该目录服务器实例维护的任何其他数据库）。

例如：

```
# db2bak

Back up directory: /var/lib/dirsrv/slapd-pki1/bak/pki1-2020_11_05_11_20_21
```

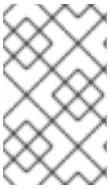
注意

当 db2bak 命令以 dirsrv 用户身份运行时，目标目录必须可由 dirsrv 写入。运行不带任何参数的命令会在 /var/lib/dirsrv/slapd- <instance_name> /bak 文件夹中创建备份，其中 db2bak 具有正确的写入权限。

要使用 bak2db 恢复 LDIF 文件，请参阅 第 13.8.1.2.2 节 “使用 bak2db 恢复”。

13.8.1.2. 恢复 LDAP 内部数据库

根据您备份 Directory 服务器实例的方式，使用 ldif2db 或 bak2db 及对应的文件来恢复数据库。



注意

在恢复数据库前，请确定停止该实例。

13.8.1.2.1. 使用 ldif2db 恢复

如果您使用 db2ldif 创建 LDIF 文件，请停止 Directory 服务器实例，并使用 ldif2db 命令导入文件。您可以指定要从备份中恢复的单个数据库。例如：

1. **停止 Directory 服务器实例：**

```
# systemctl stop dirsrv@instance_name
```

2. **为 -n 选项指定的子系统导入 -i 选项指定的文件：**

```
# ldif2db -V -n pki-tomcat-CA -i /var/lib/dirsrv/slapd-pki1/ldif/pki-ca-backup.ldif
importing data ...
[06/Nov/2020:09:27:07.103094925 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[06/Nov/2020:09:27:07.118712207 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
……...
[06/Nov/2020:09:27:09.213947960 -0500] - INFO - import_main_offline - import pki-tomcat-
```

```
CA: Closing files...
[06/Nov/2020:09:27:09.470742715 -0500] - INFO - dblayer_pre_close - All database threads
now stopped
[06/Nov/2020:09:27:09.479321728 -0500] - INFO - import_main_offline - import pki-tomcat-
CA: Import complete.  Processed 67 entries in 2 seconds. (33.50 entries/sec)
```

3. 启动 **Directory** 服务器实例：

```
# systemctl start dirsrv@instance_name
```

13.8.1.2.2. 使用 bak2db 恢复

如果您使用 **db2bak** 创建备份文件，请停止 **Directory** 服务器并使用 **bak2db** 命令导入文件；您可以指定单个数据库从备份中恢复。例如：

1. 停止 **Directory** 服务器实例：

```
# systemctl stop dirsrv@instance_name
```

2. 为 **-n** 选项指定的子系统导入文件：

```
# bak2db /var/lib/dirsrv/slapd-pki1/bak/pki1-2020_11_06_09_40_21/ -n pki-tomcat-CA -V
[06/Nov/2020:09:41:02.984808879 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[06/Nov/2020:09:41:02.991860094 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
......
[06/Nov/2020:09:41:12.853686475 -0500] - INFO - dblayer_copy_directory - Restoring file 40
(/var/lib/dirsrv/slapd-pki1/db/pki-tomcat-CA/seeAlso.db)
[06/Nov/2020:09:41:12.873881494 -0500] - WARN - dblayer_start - DB already started.
[06/Nov/2020:09:41:12.883966616 -0500] - INFO - dblayer_pre_close - All database threads
now stopped
[06/Nov/2020:09:41:12.888381193 -0500] - INFO - dblayer_restore -  Removing staging area
/var/lib/dirsrv/slapd-pki1/db/../fribak.
```

您也可以使用 命令（不使用 **-n** 选项）从备份中恢复完整的数据库。例如：

```
# bak2db /var/lib/dirsrv/slapd-pki1/bak/pki1-2020_11_06_09_40_21/ -V
[06/Nov/2020:09:53:01.977785135 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[06/Nov/2020:09:53:01.994426925 -0500] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
.........
[06/Nov/2020:09:53:02.800340285 -0500] - INFO - dblayer_restore - Restoring file 68
```

```
(/var/lib/dirsrv/slapd-pki1/db/DBVERSION)
[06/Nov/2020:09:53:02.814235053 -0500] - INFO - dblayer_copyfile - Copying
/var/lib/dirsrv/slapd-pki1/bak/pki1-2020_11_06_09_40_21/DBVERSION to
/var/lib/dirsrv/slapd-pki1/db/DBVERSION
[06/Nov/2020:09:53:03.317071092 -0500] - INFO - dblayer_pre_close - All database threads
now stopped
```

3. 启动 **Directory 服务器实例：**

```
# systemctl start dirsrv@instance_name
```

13.8.2. 备份和恢复实例目录

实例目录具有子系统实例的所有配置信息，因此备份实例目录会保留不包含在内部数据库中的配置信息。

注意

在备份实例或安全数据库前，停止子系统实例。

1. **停止子系统实例。**

```
systemctl stop pki-tomcatd@instance_name.service
```

2. **将目录保存到压缩文件中：**

```
# cd /var/lib/pki/
# tar -chvf /export/archives/pki/instance_name.tar instance_name/
```

例如：

```
# cd /var/lib/pki/
# tar -chvf /tmp/test.tar pki-tomcat/ca/
pki-tomcat/ca/
pki-tomcat/ca/registry/
pki-tomcat/ca/registry/ca/
.....
```

3. *重启子系统实例。*

```
systemctl start instance_name
```

您可以使用 CertificateCertificate Systemnbsp;System backup files（别名 数据库备份和完整实例目录备份）替换当前目录（如果数据损坏或硬件已损坏）。要恢复数据，使用 解压缩或 tar 工具解压缩存档文件，并通过现有文件复制存档。

恢复实例目录：

1. *解压缩存档：*

```
cd /export/archives/pki/
tar -xvf instance_name.tar
```

例如：

```
# cd /tmp/
# tar -xvf test.tar
pki-tomcat/ca/
pki-tomcat/ca/registry/
pki-tomcat/ca/registry/ca/
pki-tomcat/ca/registry/ca/default.cfg
.........
```

2. *如果子系统实例尚未停止，则停止它。*

```
systemctl stop pki-tomcatd@instance_name.service
```

3. *复制存档的文件以恢复实例目录：*

```
cp -r /export/archives/pki/instance_name /var/lib/pki/instance_name
```

例如：

```
# cp -r /tmp/pki-tomcat/ca/ /var/lib/pki/pki-tomcat/ca/
```

4. *重启子系统实例。*

```
systemctl start pki-tomcatd@instance_name.service
```

13.9. 运行自助测试

CertificateCertificate Systemnbsp;System 增加了允许自我证明服务器的功能。自我证明在开始时运行，也可以根据需要运行。当服务器启动时，启动自我证明运行并保持服务器启动（如果关键自我测试失败）。单击子系统控制台中的 self-tests 按钮，即可运行 on-demand self-tests。

13.9.1. 运行自助测试

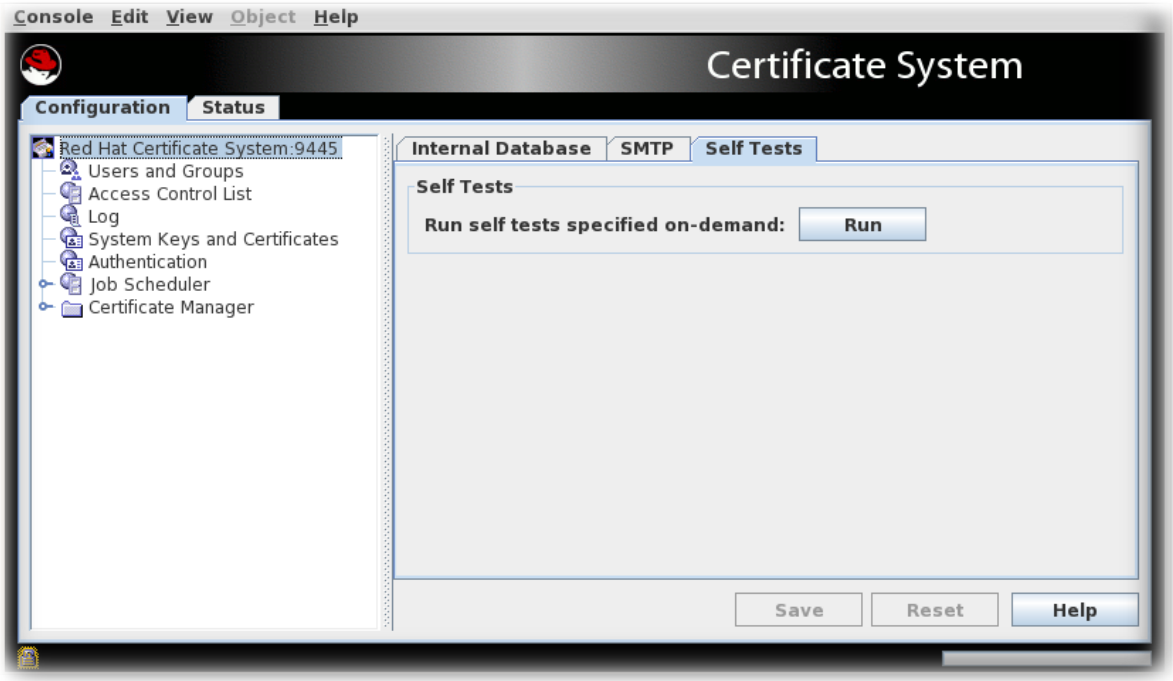
在控制台中运行 CA、OCSP、KRA 或 TKS 子系统的按需自我测试。TPS 系统的 on-demand self-tests 从 web 服务页面中运行。

13.9.1.1. 从控制台运行自助测试

1. *登录到控制台。*

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

2. *选择左侧窗格中顶部的子系统名称。*



3. 选择" 自助测试" 选项卡。

4. 点 Run。

为子系统配置的 self-tests 将运行。如果有任何关键自我证明失败，服务器将会停止。

5. 此时会出现 On-Demand Self Tests Results 窗口，显示针对这个结果的已记录事件。

13.9.1.2. 运行 TPS Self-Tests

使用命令行界面(CLI)运行 TPS 自助演示：

- pki tps-selftest-find
- pki tps-selftest-run
- pki tps-selftest-show

13.9.2. 自我测试日志记录

一个单独的日志 selftest.log 添加到包含启动自tests 和 on-demand self-tests 的报告的日志目录中。通过更改 CS.cfg 文件中的日志的设置来配置此日志。详情请参阅 红帽证书系统规划、安装和部署指南中的 修改自助测试配置 部分。

13.9.3. 配置 POSIX 系统 ACL

POSIX 系统访问控制规则为系统用户权限提供精细粒度。必须在完全配置后为每个实例设置这些 ACL。有关 ACL 的详情，请查看 Red Hat Enterprise Linuxnbsp;Hat Enterprise Linuxnbsp 中对应的章节；Linux 存储管理指南。

13.9.3.1. 为 CA、KRA、OCSP、TKS 和 TPS 设置 POSIX 系统 ACL

现代文件系统（如 ext4 和 XFS）默认启用 ACL，最有可能在现代 Red Hat Enterprise Linuxnbsp;Hat Enterprise Linuxnbsp;Linux 安装中使用。

1. **停止该实例。**

```
systemctl stop pki-tomcatd@instance_name.service
```

2. **将实例的目录和文件的 pkiadmin 组设置为 pkiadmin 组。**

```
# setfacl -R -L -m g:pkiadmin:r,d:g:pkiadmin:r /var/lib/pki/instance_name
```

3. **在所有目录中应用执行(x)ACL 权限：**

```
# find -L /var/lib/pki/instance_name -type d -exec setfacl -L -n -m
g:pkiadmin:rx,d:g:pkiadmin:rx {} \;
```

4. **从实例的 signedAudit/ 目录及其相关文件中删除 pkiadmin 组的组可读性：**

```
# setfacl -R -L -x g:pkiadmin,d:g:pkiadmin /var/lib/pki/instance_name/logs/signedAudit
```

5. **为实例的 signedAudit/ 目录及其相关文件设置 pkiaudit 组的组可读性：**

```
# setfacl -R -L -m g:pkiaudit:r,d:g:pkiaudit:r /var/lib/pki/instance_name/logs/signedAudit
```

6. **重新对签名Audit/ 目录及其所有子目录重新执行(x)ACL 权限：**

```
# find -L /var/lib/pki/instance_name/logs/signedAudit -type d -exec setfacl -L -n -m
g:pkiaudit:rx,d:g:pkiaudit:rx {} \;
```

7. **启动实例。**

```
systemctl start pki-tomcatd@instance_name.service
```

8. **使用 getfacl 命令显示当前 ACL 设置，确认文件访问控制是否已正确应用：**

```
# getfacl /var/lib/pki/instance_name
/var/lib/pki/instance_name/subsystem_type/logs/signedAudit/
```

```
getfacl: Removing leading '/' from absolute path names
# file: var/lib/pki/instance_name
# owner: pkiuser
# group: pkiuser
user::rwx
group::rwx
group:pkiadmin:r-x
mask::rwx
other::r-x
default:user::rwx
default:group::rwx
default:group:pkiadmin:r-x
default:mask::rwx
default:other::r-x

# file: var/lib/pki/instance_name/logs/signedAudit
# owner: pkiuser
# group: pkiaudit
user::rwx
group::rwx
group:pkiaudit:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:pkiaudit:r-x
default:mask::rwx
default:other::---
```

第 14 章 管理证书证书系统启动; 系统用户和组

本章介绍了如何设置授权来访问管理、代理服务和端点页面。

14.1. 关于授权

授权 是允许访问与证书证书 Systemnbsp;System 关联的特定任务的过程。访问权限可以限于允许对特定用户或组的特定子系统区域执行某些任务，以及不同的用户和组的不同任务。

用户特定于创建它们的子系统。每个子系统都有自己的一组独立于安装的任何其他子系统的用户。用户放置在组中，它们可以预定义或创建用户。通过 访问控制列表 (ACL)将权限分配给组。存在与管理控制台、代理服务接口和端点页面中相关的 ACL，它们可在允许操作继续操作前执行授权检查。在每个 ACL 中创建 访问控制指令 (ACI)，以明确允许或拒绝该 ACL 的可能操作、组或 IP 地址。

ACL 包含创建的默认组的默认 ACI 组。可以修改这些 ACI，以更改预定义组的权限，或者为新建的组分配特权。

授权通过以下过程进行：

1. 用户使用 CertificateCertificate Systemnbsp;System 用户 ID 和密码或证书向接口进行身份验证。

2. 服务器将用户 ID 和密码与数据库中存储的用户 ID 和密码匹配，或者检查数据库中存储的证书来验证用户。使用基于证书的验证时，服务器还会检查证书是否有效，并通过将证书的 DN 与用户条目关联并查找用户的组成员资格。使用基于密码的身份验证时，服务器会根据用户 ID 检查密码，然后通过将用户 ID 与组中包含的用户 ID 关联来查找用户的组成员资格。

3. 当用户试图执行操作时，授权机制会比较用户 ID、用户所属的组，或者用户所属的 IP 地址，或者该用户的 IP 地址针对该用户、组或 IP 地址。如果存在允许该操作的 ACL，则操作继续进行。

14.2. 默认组

用户的特权由用户的 group(role)成员身份决定。用户可以分配给以下三个组（角色）：

- 管理员。此组被授予对管理界面中所有可用任务的完全访问权限。

- 代理。此组被授予对代理服务接口中所有可用任务的完全访问权限。

- 审核员.通过此组查看签名的审计日志，可以访问此组。这个组没有任何其他特权。

有一个专为子系统之间的通信而创建的第四个角色。管理员绝对不应将真实用户分配给这样的角色：

- 企业管理员。在配置期间，每个子系统实例将自动分配一个特定于子系统的角色，作为企业管理员。这些角色在安全域的子系统之间自动提供可信关系，这样每个子系统都可以高效地与其他子系统交互。

14.2.1. 管理员

管理员具有执行所有管理任务的权限。用户通过将 添加到组的 Administrators 组来指定或识别为管理员。该组的每个成员均具有证书证书系统nbsp;System 的管理权限。

必须至少为每个证书证书系统nbsp;System 实例定义了一个管理员，但实例可以拥有的管理员数量没有限制。配置实例时会创建第一个管理员条目。

管理员使用其证书证书系统nbsp 并通过简单的绑定进行身份验证；系统用户 ID 和密码。

表 14.1. 安全域用户角色

角色	描述
安全域管理员	• 在安全域的用户和组数据库中添加和修改用户。 • 管理共享信任策略。 • 管理域服务中的访问控制。 默认情况下，托管域的 CA 管理员将作为安全性域管理员分配。

角色	描述
Enterprise CA 管理员	• 从域中的任何 CA 自动批准任何子 CA、服务器和子系统证书。 • 在安全域中注册和取消注册 CA 子系统信息。
企业 KRA 管理员	• 从域中的任何 CA 自动批准任何传输、存储、服务器和子系统证书。 • 在安全域中注册和取消注册 KRA 子系统信息。 • 将 KRA 连接器信息推送到任何 CA。
Enterprise OCSP 管理员	• 从域中的任何 CA 自动批准所有 OCSP、服务器和子系统证书。 • 在安全域中注册和取消注册 OCSP 子系统信息。 • 将 CRL 发布信息推送到任何 CA。
Enterprise TKS 管理员	• 从域中的任何 CA 自动批准任何服务器和子系统证书。 • 在安全域中注册和取消注册 TKS 子系统信息。
Enterprise TPS 管理员	• 从域中的任何 CA 自动批准任何服务器和子系统证书。 • 在安全域中注册和取消注册 TPS 子系统信息。

必要时，安全域管理员可以管理安全域和各个子系统的访问控制。例如，安全域管理员可以限制访问，使只有财务部门 KRA 管理员才能设置财务部门 KRA。

企业子系统管理员具有足够的特权，以便对域中的子系统执行操作。例如，企业 CA 管理员在配置过程中自动批准了子 CA 证书的权限。或者，如果必要，安全域管理员可以限制此限制。

14.2.2. 审核员

审核员可以查看签名的审计日志，并被创建来审核系统的操作。审核员无法以任何方式管理服务器。

创建审核员方法是将用户添加到 Auditors 组，并将 auditor 的证书存储在用户条目中。auditor 的证书用于加密用于为审计日志签名的密钥对的私钥。

在配置子系统时，将设置 Auditors 组。配置期间不会将审核员分配给此组。

审核员通过一个简单的绑定（使用其 UID 和密码）在管理控制台中进行身份验证。经过身份验证后，审核员只能查看审计日志。不能编辑系统的其他部分。

14.2.3. 代理

代理是分配了终端证书和密钥管理权限的用户。代理可以访问代理服务接口。

可通过将用户分配给适当的子系统代理组来创建代理，并确定代理必须用于 SSL 客户端请求的证书，以便其供代理为服务请求。每个子系统都有自己的代理组：

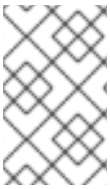
- 证书管理器代理组。
- 密钥恢复授权代理组。
- Online Certificate Status Agents 组。
- 令牌密钥服务代理组。
- 令牌处理系统代理组。

每个证书证书系统nbsp;System 子系统具有自己的代理，其角色由子系统定义。每个子系统必须至少有一个代理，但对子系统可以有的代理数量没有限制。

CertificateCertificate Systemnbsp;System 通过在其内部数据库中检查用户的 SSL 客户端证书来识

别和验证具有代理特权的用户。

14.2.4. 企业级组



注意

应该不会向该组分配真实用户。

在子系统配置期间，每个子系统实例都加入到一个安全域。每个子系统实例将自动分配一个特定于子系统的角色，作为企业管理员。这些角色在安全域的子系统之间自动提供可信关系，这样每个子系统都可以高效地与其他子系统交互。例如，这允许 OCSP 将 CRL 发布信息推送到域中的所有 CA，KRAs 推送 KRA 连接器信息，以及 CA 可以自动批准 CA 中生成的证书。

企业子系统管理员具有足够的特权，以便对域中的子系统执行操作。每个子系统都有自己的安全域角色：

- Enterprise CA 管理员
- 企业 KRA 管理员
- Enterprise OCSP 管理员
- Enterprise TKS 管理员
- Enterprise TPS 管理员

另外，CA 实例有一个安全域管理员组，用于管理域、访问控制、用户和域内的信任关系。

每个子系统管理员使用 SSL 客户端证书和由安全域 CA 配置期间发布的子系统证书，向其他子系统进行身份验证。

14.3. 管理 CA、OCSP、KRA 或 TKS 的用户和组

许多用户可以执行的操作由他们所属的组指定；例如，CA 的代理管理证书和配置集，而管理员管理 CA 服务器配置。

四个子系统 - CA、OCSP、KRA 和 TKS - 使用 Java 管理控制台管理组和用户。TPS 有基于 Web 的管理服务，用户和组通过 Web 服务页面配置。

14.3.1. 管理组

14.3.1.1. 创建新组

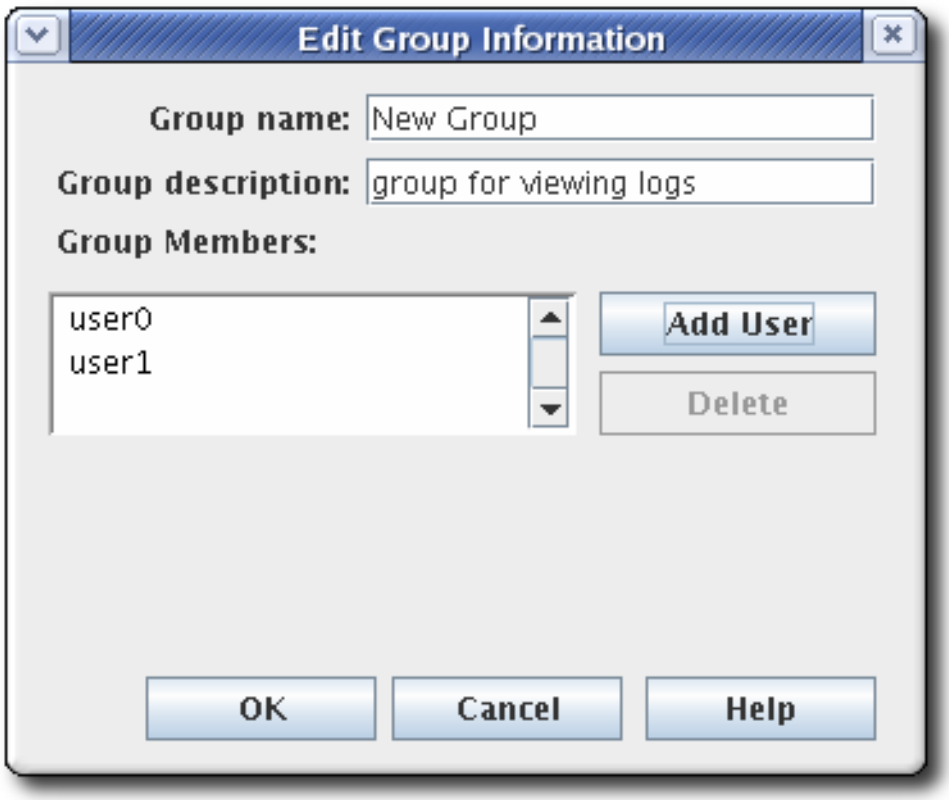
1. 登录到管理控制台。

```
pkiconsole https://server.example.com:8443/subsystem_type
```

2. 从左侧的导航菜单中选择" 用户和组 "。

3. 选择 Groups 选项卡。

4. 点 Edit，然后填写组信息。



只能添加已存在于内部数据库中的用户。

5. 编辑 ACL 以授予组特权。如需更多信息，请参阅 第 14.5.4 节 “编辑 ACL”。如果没有在组的 ACL 中添加 ACl，组将无法访问任何部分 CertificateCertificate Systemnbsp;System。

14.3.1.2. 更改组中的成员

可从所有组中添加或删除成员。管理员的组必须至少有一个用户条目。

1. 登录到管理控制台。
2. 从左侧的导航树中选择" 用户和组 "。
3. 点 Groups 标签页。
4. 从名称列表中选择组，再点击 Edit。
5. 进行适当的更改。

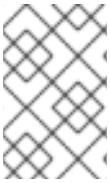
   - 要更改组描述，在 Group description 字段中输入新描述。
   - 要从组中删除用户，选择用户，再单击删除。
   - 要添加用户，请单击 添加用户。从对话框中选择要添加的用户，然后单击确定。

14.3.2. 管理用户（管理员、代理和审核员）

每个子系统的用户会单独维护。仅仅因为一个子系统中的管理员并不知道对另一个子系统具有任何权利（甚至是用户条目）。可以为子系统配置用户，并使用其用户证书、信任的代理、管理员或审核员来配置。

14.3.2.1. 创建用户

安装证书系统后，只有在设置期间创建的用户就存在。这部分论述了如何创建其他用户。



注意

出于安全考虑，为证书系统用户创建单独的帐户。

14.3.2.1.1. 使用命令行创建用户

使用命令行创建用户：

1. 添加用户帐户。例如，将 示例 用户添加到 CA：

```
# pki -d ~/.dogtag/pki-instance_name/ca/alias/ -c password -n caadmin \
    ca-user-add example --fullName "Example User"
---------------------
Added user "example"
---------------------
  User ID: example
  Full name: Example User
```

此命令使用 caadmin 用户添加新帐户。

2. （可选）将用户添加到组中。例如，将 示例 用户添加到 证书管理器代理 组中：

```
# pki -d ~/.dogtag/pki-instance_name/ -p password -n "caadmin" \
    user-add-membership example Certificate Manager Agents
```

3. 创建证书请求：

- 如果您的证书系统环境中存在密钥恢复授权(KRA)：

```
# CRMFPopClient -d ~/.dogtag/pki-instance_name/ -p password \
    -n "user_name" -q POP_SUCCESS -b kra.transport -w
"AES/CBC/PKCS5Padding" \
    -v -o ~/user_name.req
```

此命令在 ~/user_name.req 文件中以 CRMF 格式存储证书签名请求(CSR)。

- 如果您的证书系统环境中不存在密钥恢复授权(KRA)：

```
# PKCS10Client -d ~/.dogtag/pki-instance_name/ -p password \
  -n "user_name" -o ~/user_name.req
```

此命令将 CSR 以 pkcs10 格式存储在 ~/user_name.req 文件中。

4. 创建注册请求：

a. 使用以下内容创建 ~/cmc.role_crmf.cfg 文件：

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
#Multiple files are supported. They must be separated by space.
input=~/user_name.req

#output: full path for the CMC request in binary format
output=~/cmc.role_crmf.req

#tokenname: name of token where agent signing cert can be found (default is
internal)
tokenname=internal

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
nickname=PKI Administrator for Example.com

#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=~/.dogtag/pki-instance_name/

#password: password for cert8.db which stores the agent
#certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf
```

根据您的环境和上一步中使用的 CSR 格式设置参数。

b.

将之前创建的配置文件传递给 **CMCRequest** 工具，以创建 **CMC** 请求：

```
# CMCRequest ~/cmc.role_crmf.cfg
```

5.

通过 **CMS** 提交证书管理**(CMC)**请求：

a.

使用以下内容创建 **~/HttpClient_role_crmf.cfg** 文件：

```
# #host: host name for the http server
host=server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in binary format
input=~/cmc.role_crmf.req

#output: full path for the response in binary format
output=~/cmc.role_crmf.resp

#tokenname: name of token where SSL client authentication cert can be found
(default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=~/.dogtag/pki-instance_name/

#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert8.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=PKI Administrator for Example.com

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitCMCFull
```

根据您的环境设置参数。

b.

将请求提交到 CA：

```
# HttpClient ~/HttpClient_role_crmf.cfg
Total number of bytes read = 3776
after SSLSocket created, thread token is Internal Key Storage Token
client cert is not null
handshake happened
writing to socket
Total number of bytes read = 2523
MIIJ1wYJKoZIhvcNAQcCoIIJyDCCCcQCAQMxDzANBglghkgBZQMEAgEFADAxBg
gr
...
The response in data format is stored in ~/cmc.role_crmf.resp
```

c.

验证结果：

```
# CMCResponse ~/cmc.role_crmf.resp
Certificates:
   Certificate:
      Data:
         Version:  v3
         Serial Number: 0xE
         Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
         Issuer: CN=CA Signing Certificate,OU=pki-instance_name Security Domain
         Validity:
            Not Before: Friday, July 21, 2017 12:06:50 PM PDT America/Los_Angeles
            Not  After: Wednesday, January 17, 2018 12:06:50 PM PST
America/Los_Angeles
         Subject: CN=user_name
...
Number of controls is 1
Control #0: CMCStatusInfoV2
  OID: {1 3 6 1 5 5 7 7 25}
  BodyList: 1
  Status: SUCCESS
```

6.

（可选）将证书作为用户导入到自己的 ~/.dogtag/pki-instance_name/ 数据库：

```
# certutil -d ~/.dogtag/pki-instance_name/ -A -t "u,u,u" -n "user_name certificate" -i
~/cmc.role_crmf.resp
```

7.

在用户记录中添加证书：

a.

列出用户发布的证书，以发现证书的序列号。例如，列出在证书主体中包含 示例 用户名的证书：

```
pki -d ~/.dogtag/pki-instance_name/ -c password -n caadmin ca-user-cert-find
example
-----------------
1 entries matched
-----------------
  Cert ID: 2;6;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI
Administrator,E=example@example.com,O=EXAMPLE
  Version: 2
  Serial Number: 0x6
  Issuer: CN=CA Signing Certificate,O=EXAMPLE
  Subject: CN=PKI Administrator,E=example@example.com,O=EXAMPLE
----------------------------
Number of entries returned 1
```

下一步中需要证书的序列号。

b.

使用证书仓库中的序列号将证书添加到证书系统数据库中的用户帐户。例如，对于 CA 用户：

```
pki -c password -n caadmin ca-user-cert-add example --serial 0x6
```

14.3.2.1.2. 使用控制台创建用户

使用 PKI 控制台创建用户：

1.

登录到管理控制台。

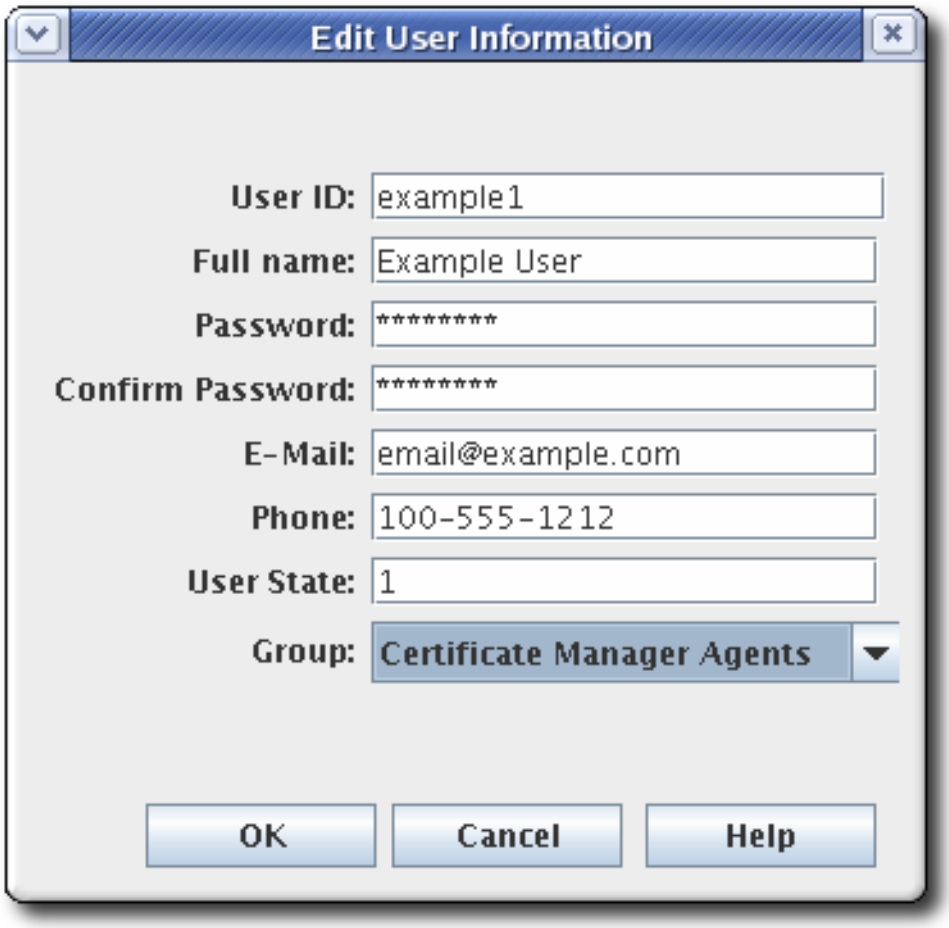
```
pkiconsole https://server.example.com:8443/subsystem_type
```

2.

在 Configuration 选项卡中，选择" 用户和组 "。点 Add。

3.

在编辑用户信息对话框中填写信息。



大多数信息都是标准用户信息，如用户名、电子邮件地址和密码。此窗口还包含一个名为 User State 的字段，它可以包含任何字符串，用于添加关于用户的附加信息；最重要的是，此字段可以显示为活动用户。

4. 选择用户所属的组。用户的组成员资格决定了用户具有什么权限。将代理、管理员和审核员分配到适当的子系统组。

5. 存储用户的证书。

   a. 通过 CA 最终用户服务页面请求用户证书。

   b. 如果没有为用户配置集配置自动注册，则批准证书请求。

   c. 使用通知电子邮件中提供的 URL 检索证书，并将 base-64 编码证书复制到本地文件或剪贴板。

d.
选择新用户条目，然后单击 证书。

e.
单击 Import，并粘贴到 base-64 编码证书中。

14.3.2.2. 更改证书证书系统启动;系统用户的证书

1. 登录到管理控制台。
2. 选择 "用户和组 "。
3. 从用户 ID 列表中选择要编辑的用户，然后点 Certificates。
4. 单击 Import 以添加新证书。
5. 在 Import Certificate 窗口中，将新证书粘贴到文本区域中。包含 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE----- 标记行。

14.3.2.3. 续订管理员、代理和审核员用户证书

更新证书的方法有两种。重新生成证书 会提取其原始密钥及其原始配置集并请求，并重新创建相同的密钥，其中包含新的有效期期限和过期日期。重新打包 证书，重新提交到原始配置集的初始证书请求，但生成新的密钥对。管理员可以通过重新密钥来续订管理员证书。

每个子系统都有一个 bootstrap 用户，可在创建子系统时创建。通过使用默认续订配置集之一，可以为此用户请求一个新证书。

管理员用户的证书可以在最终用户注册表单中直接续订，使用原始证书的序列号。

1. 在 CA 的最终用户表单中续订 admin 用户证书，如 第 5.5.1.1.2 节 “基于证书的续订” 所述。这必须与第一次签发证书（或它的克隆）相同。

可使用末尾实体页面中基于证书的续订表格来续订代理证书。自助服务 SSL 客户端证书.此表单可识别和更新保存在浏览器的证书存储中直接存储的证书。

注意

您还可以使用 certutil 更新证书，如 第 16.3.3 节 “使用 certutil 更新证书” 所述。certutil 使用带有原始密钥的输入文件，而不是使用浏览器中存储的证书来启动续订。

2. 将更新的用户证书添加到内部 LDAP 数据库中的用户条目。

a. 打开子系统的控制台。

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

b. 配置 | 用户和组群 | 用户 | 管理 | 证书 | 导入

c. 在 Configuration 选项卡中，选择" 用户和组 "。

d. 在 Users 选项卡中，使用更新的证书双击用户条目，然后单击 证书。

e. 单击 Import，并粘贴到 base-64 编码证书中。

这可以通过使用 ldapmodify 将更新的认证直接添加到内部 LDAP 数据库中的用户条目，方法是替换用户条目中的 userCertificate 属性，如 uid=admin,ou=body,dc=subsystem -base-DN。

14.3.2.4. 删除证书证书系统启动; 系统用户

可以从内部数据库删除用户。从内部数据库中删除用户将从该用户所属的所有组中删除。要从特定组中删除用户，请修改组成员资格。

通过以下操作从内部数据库中删除特权用户：

1. 登录到管理控制台。

2. 从左侧的导航菜单中选择" 用户和组 "。

3. 从用户 ID 列表中选择用户，然后点 Delete。

4. 提示时确认删除。

14.4. 为 TPS 创建和管理用户

TPS 用户定义了三个 角色，它充当 TPS 的组：

- 代理，用于执行实际令牌管理操作，如设置令牌状态和更改令牌策略
- 管理员，负责管理 TPS 子系统的用户，并对令牌进行有限控制
- Operator 没有管理控制，但可以查看和列出通过 TPS 执行的令牌、证书和活动

无法为 TPS 添加其他组。

所有 TPS 子系统用户都针对包含其证书的 LDAP 目录数据库进行身份验证（因为访问 TPS 的 Web 服务需要基于证书的身份验证），并且验证过程会检查 TPS 组条目 - ou=TUS Agents、ou=TUS Administrators、ou=TUS Operators、ou=TUS Operators、ou=TUS Operators 使用 httpd_token 模块。

通过 Web UI 或 CLI 添加和管理 TPS 的用户。Web UI 可通过 https://server.example.com:8443/tps/ui/ 访问。

要使用 Web UI 或 CLI，您必须使用用户证书进行身份验证。

14.4.1. 列出和搜索用户

14.4.1.1. 通过 Web UI

从 Web UI 列出用户：

1. 点 Accounts 选项卡。
2. 点 Users 菜单项。用户列表会出现在页面上。
3. 要搜索某些用户，请在搜索字段中输入关键字并按 Enter 键。要再次列出所有用户，请删除关键字并按 Enter。

14.4.1.2. 从命令行

要从 CLI 列出用户，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-find
```

要从 CLI 查看用户详情，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-show username
```

14.4.2. 添加用户

14.4.2.1. 通过 Web UI

从 Web UI 添加用户：

1. 点 Accounts 选项卡。
2. 点 Users 菜单项。
3. 点 Users 页面中的 Add 按钮。

4.
填写用户 ID、全名和 TPS 配置集。

5.
点 Save 按钮。

14.4.2.1.1. 从命令行

要从 CLI 添加用户，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-add username --fullName full_name
```

14.4.3. 为用户设置配置集

TPS 配置集与 CA 配置集非常相似，它定义了用于处理不同类型的令牌的规则。该配置集根据令牌的某些特征自动分配给令牌，如 CUID。用户只能查看分配给它们的配置集的令牌。

注意

用户只能查看与为其配置的配置集相关的条目，包括令牌操作和令牌本身。如果管理员能够搜索和管理在 TPS 中配置的所有令牌，管理员用户条目应设置为 All profiles。为用户设置特定配置集是一个简单的方法，可以对特定用户或令牌类型控制 operator 和代理的访问。

令牌配置集是应用到令牌的策略和配置的集合。令牌配置文件根据令牌本身中的某种属性自动映射到令牌，如 CCUID 范围。令牌配置集作为 CA 配置集目录中的其他证书配置集创建（如 Red Hat Certificate System planning、Installing 和 Deployment Guide），然后添加到 TPS 配置文件 CS.cfg 中，将 CA 的令牌配置集映射到令牌类型。配置令牌映射包括在 第 6.7 节 “映射解析器配置” 中。

从 Web UI 管理用户配置集：

1.
点 Accounts 选项卡。

2.
点 Users 菜单项。

3.
点您要修改的用户的用户名。

4.
点 Edit 链接。

5.
在 TPS Profile 字段中输入由逗号分开的配置集名称，或者输入 All Profiles。

6.
点 Save 按钮。

14.4.4. 管理用户角色

角色只是 TPS 中的组。每个角色可以查看 TPS 服务页面的不同标签页。组可编辑，因此可以为用户添加或删除角色分配。

用户可以属于多个角色或组。例如，bootstrap 用户属于所有三个组。

14.4.4.1. 通过 Web UI

从 Web UI 管理组成员：

1.
点 Accounts 选项卡。

2.
点 Groups 菜单项。

3.
点您要更改的组名称，如 TPS Agents。

4.
将用户添加到此组中：

a.
点击 Add 按钮。

b.
输入用户 ID。

c.
点击 Add 按钮。

5.
从这个组中删除用户：

a.
选中用户旁边的复选框。

b.
点 删除按钮。

c.
点确定按钮。

14.4.4.2. 从命令行

要从 CLI 列出组，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-find
```

要从 CLI 列出组成员，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-find
group_name
```

要从 CLI 将用户添加到组中，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-add
group_name user_name
```

要从 CLI 中的组中删除用户，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-del
group_name user_name
```

14.4.5. 管理用户角色

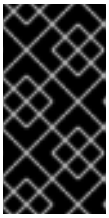
用户证书可以通过 CLI 管理：

- 要列出用户证书，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-find
user_name
```

- 向用户添加证书：

1. 获取新用户的用户证书。请求和提交证书在 第 5 章 请求、注册和管理证书 中进行了说明。



重要

TPS 管理员必须具有签名证书。推荐的配置集是手动用户签名和加密证书注册。

2. 运行以下命令：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-add
user_name --serial cert_serial_number
```

- 要从用户中删除证书，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-del
user_name cert_id
```

14.4.6. 更新 TPS 代理和管理员证书

重新生成证书会提取其原始密钥及其原始配置集并请求，并重新创建相同的密钥，其中包含新的有效期期限和过期日期。

TPS 创建了一个 bootstrap 用户，可在创建子系统时创建。当使用默认续订配置集之一时，可以为这个用户请求一个新证书。

管理员用户的证书可以在最终用户注册表单中直接续订，使用原始证书的序列号。

1. 通过 CA 的最终用户表单续订用户证书，如 第 5.5.1.1.2 节 “基于证书的续订” 所述。这必须与第一次签发证书（或它的克隆）相同。

通过使用最终实体页面中的基于证书的续订表格（自助式 用户 SSL 客户端证书）可以续订代理证书。此表单可识别和更新保存在浏览器的证书存储中直接存储的证书。

注意

您还可以使用 certutil 更新证书，如 第 16.3.3 节 “使用 certutil 更新证书” 所述。certutil 使用带有原始密钥的输入文件，而不是使用浏览器中存储的证书来启动续订。

2. 向用户添加新证书并删除旧证书，如 第 14.4.5 节 “管理用户角色” 所述。

14.4.7. 删除用户

警告

可以删除 最后一个 用户帐户，操作无法撤消。请非常小心，请告知要删除的用户。

从 Web UI 中删除用户：

1. 点 Accounts 选项卡。
2. 点 Users 菜单项。
3. 选中要删除的用户旁边的复选框。
4. 点 删除按钮。
5. 点确定按钮。

要从 CLI 删除用户，请运行：

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-del user_name
```

14.5. 为用户配置访问控制

授权 是检查用户是否允许执行操作的机制。授权点在需要授权检查的某些组操作中定义。

14.5.1. 关于访问控制

访问控制列表 (ACL)是指定对服务器操作的授权的机制。每个操作集合存在 ACL，其中进行授权检查。可将其他操作添加到 ACL。

ACL 包含 访问控制指令 (ACI)，它们专门允许或拒绝操作，如读或修改。ACI 还包含 evaluator 表达式。ACL 的默认实施仅以可能的 evaluator 类型指定用户、组和 IP 地址。ACL 中的每个 ACI 指定是否允许或拒绝访问、允许或拒绝特定 Operator，以及哪些用户、组或 IP 地址是被允许还是拒绝执行该操作。

CertificateCertificate Systemnbsp 的特权；通过更改与用户所属组的访问控制列表(ACL)更改系统用户，或者用于用户的 IP 地址。通过在访问控制列表中添加该组，为新组分配访问控制。例如，对于仅有权查看日志的管理员的新组，LogAdmins 则可添加到与日志相关的 ACL 中，以允许读取或修改对此组的访问权限。如果这个组没有添加到任何其他 ACL 中，则此组成员只能访问日志。

通过编辑 ACL 中的 ACI 条目来更改用户、组或 IP 地址的访问权限。在 ACL 界面中，每个 ACI 都会显示在其自己的行中。在这个接口窗口中，ACI 具有以下语法：

```
allow|deny (operation) user|group|IP="name"
```

注意

IP 地址可以是 IPv4 或 IPv6 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF

例如，以下是 ACI，允许管理员执行读取操作：

```
allow (read) group="Administrators"
```

ACI 可以配置多个操作或操作。操作以逗号分开，其中两侧没有空格。例如：

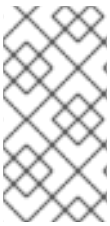
```
allow (read,modify) group="Administrators"
```

ACI 可以有多个组、用户或 IP 地址，方法是将它们分隔为两个管道符号(||)，并在两侧面上有一个空格。例如：

```
allow (read) group="Administrators" || group="Auditors"
```

管理控制台可以创建或修改 ACI。接口设置是否允许或拒绝 Allow 和 Deny 字段中的操作，设置 Operations 字段中可以执行的操作，然后在 syntax 字段中列出组、用户或 IP 地址被授权或拒绝访问。

ACI 可以允许或拒绝指定组、用户 ID 或 IP 地址的操作。通常，不需要创建 ACI 来拒绝访问。如果没有允许 ACI 包含用户 ID、组或 IP 地址，则拒绝组、用户 ID 或 IP 地址。



注意

如果用户没有明确允许访问资源的任何操作，则此用户将被认为拒绝；他不需要拒绝访问。

例如，用户 JohnB 是 Administrators 组的成员。如果 ACL 仅具有以下 ACL，JohnB 会拒绝任何访问，因为与任何允许 ACI 不匹配：

```
Allow (read,modify) group="Auditors" || user="BrianC"
```

通常不需要包含 deny 语句。然而，在有些情况下，对于指定情况非常有用。例如，JohnB （管理员组的成员）刚刚触发。如果用户无法立即删除，可能需要拒绝对 JohnB 的访问。另一个情况是，用户 BrianC 是管理员，但他不应该更改某些资源。由于 Administrators 组必须访问此资源，因此通过创建一个拒绝这个用户访问的 ACI，可以明确拒绝 BrianC 的访问。

允许的权限是 ACI 控制的操作，可以通过允许或拒绝权限来执行操作。为 ACL 设置的操作因 ACL 和子系统而异。可以定义的两个常见操作是读取和修改。

ACI 编辑器的 syntax 字段为表达式设置 evaluator。evaluator 可以指定组、名称和 IP 地址（IPv4 和 IPv6 地址）。它们按照等于(=)或不等号(!=)设置的名称指定。

在 ACL 中包括组的语法是 group="groupname"。排除组的语法为 group!="groupname"，它允许除名为 的组外的任何组。例如：

```
group="Administrators" || group!="Auditors"
```

也可以使用正则表达式来指定组，例如使用星号(*)等通配符字符。例如：

```
group="* Managers"
```

有关支持的正则表达式模式的更多信息，请参阅 https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html。

在 ACL 中包含用户的语法是 用户="userID"。要排除用户的语法为 user!="userID"，它允许除名为的用户 ID 外的任何用户 ID 除外。例如：

```
user="BobC" || user!="JaneK"
```

要指定所有用户，请提供该人 的值。例如：

```
user="anybody"
```

也可以使用正则表达式来指定用户名，如使用通配符字符，如星号(*)。例如：

```
user="*johnson"
```

有关支持的正则表达式模式的更多信息，请参阅 https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html。

在 ACL 中包含 IP 地址的语法为 ipaddress="ipaddress"。从 ACL 中排除 ID 地址的语法为 ipaddress!="ipaddress"。IP 地址使用数字值指定；不允许使用 DNS 值。例如：

```
ipaddress="12.33.45.99"
ipaddress!="23.99.09.88"
```

IP 地址可以是 IPv4 地址，如上所示，也可以是 IPv6 地址。IPv4 地址的格式为 n.n.n.n 或

n.n.n.n,m.m.m.m （子网掩码）。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：

```
ipaddress="0:0:0:0:0:0:13.1.68.3"
```

也可以使用正则表达式来指定 IP 地址，如使用星号(*)等通配符字符。例如：

```
ipaddress="12.33.45.*"
```

有关支持的正则表达式模式的更多信息，请参阅 https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html。

通过将每个值分开两个管道字符(||)，创建一个带有多个值的字符串。例如：

```
user="BobC" || group="Auditors" || group="Administrators"
```

14.5.2. 更改 subsystem 的访问控制设置

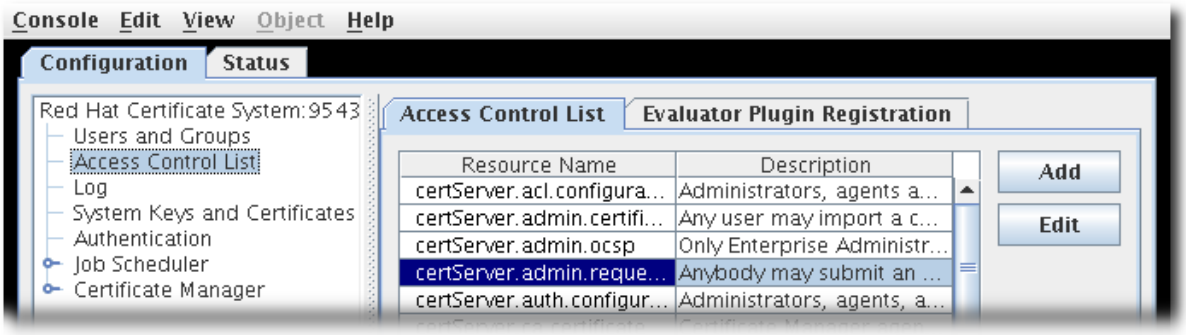
有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅 Red Hat Certificate System Planning、Installation System Planning、Installation System Planning、Installing and deployment Guide 中的 subsystem 的访问控制设置 一节。

14.5.3. 添加 ACL

ACL 存储在内部数据库中，且只能在管理控制台中修改。

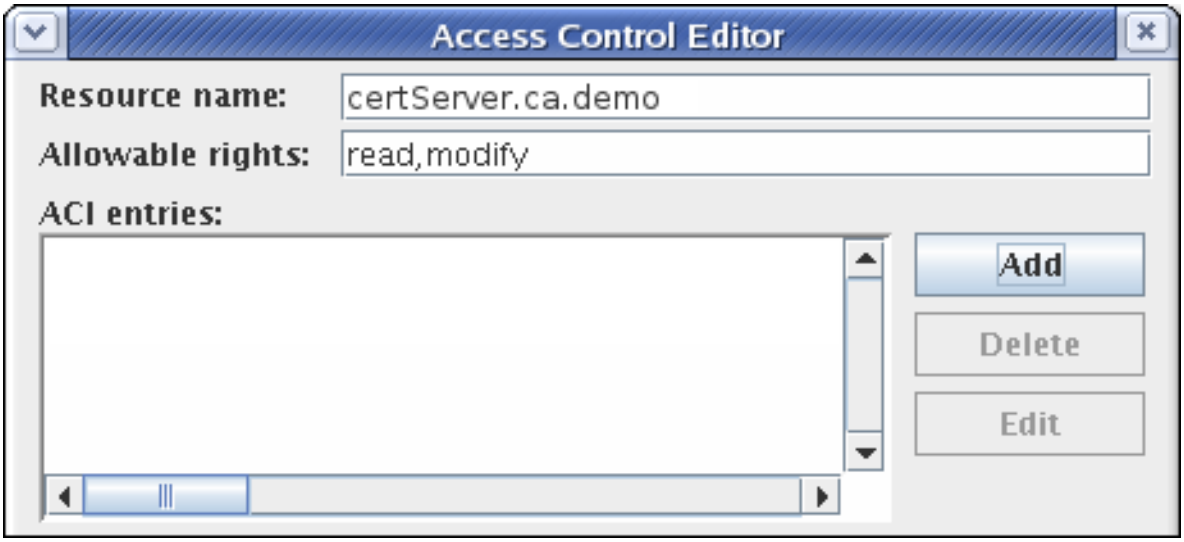
添加新 ACL：

1. 登录到管理控制台。
2. 选择 Access Control List。

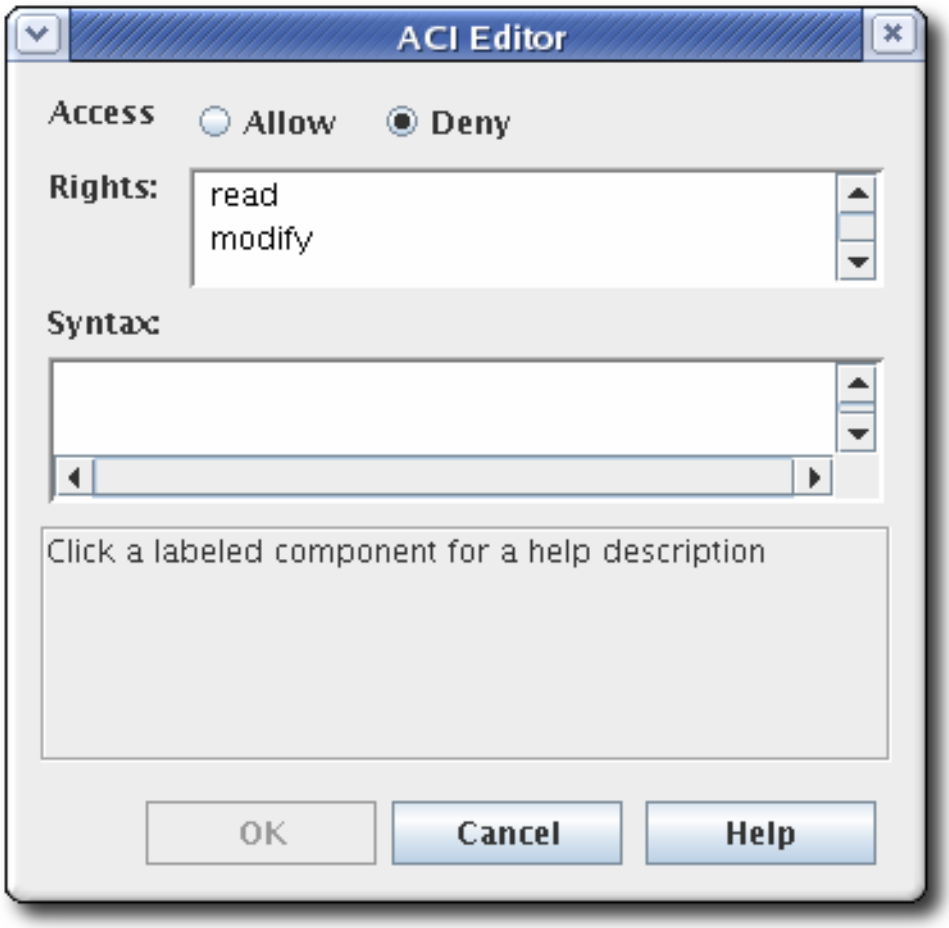


3. ***点 Add 打开 Access Control Editor。***

4. ***填写 Resource name 和 Available rights 字段。***



5. ***要添加访问控制指令(ACI)，请单击 Add 并提供 ACI 信息。***



a.

从 Access 字段中选择允许或拒绝单选按钮，以允许或拒绝指定的组、用户或 IP 地址。有关允许或拒绝访问的详情，请参考 第 14.5.1 节 “关于访问控制”。

b.

设置权利。可用的选项为 读取 和修改。要选择两者，请在选择条目时按住 Ctrl 或 Shift 按钮。

c.

指定在 syntax 字段中授予或拒绝访问的用户、组或 IP 地址。有关语法的详情，请查看 第 14.5.1 节 “关于访问控制”。

6.

点 OK 返回 Access Control Editor 窗口。

7.

单击 OK 以存储 ACI。

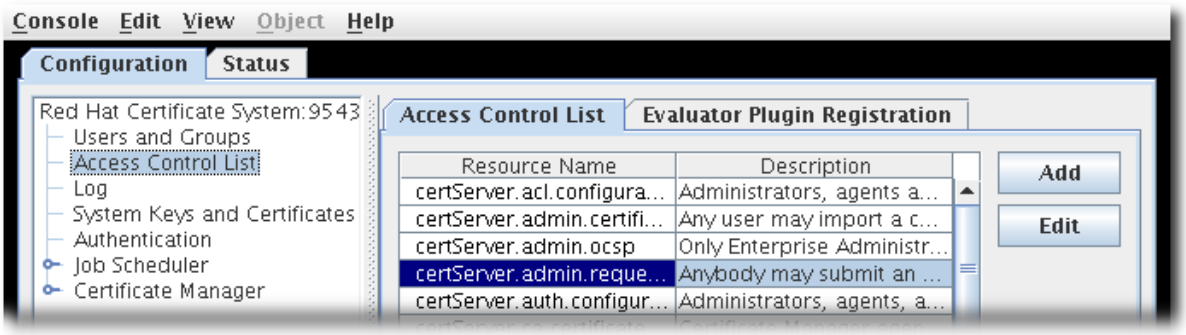
14.5.4. 编辑 ACL

ACL 存储在内部数据库中，且只能在管理控制台中修改。

编辑现有的 ACL：

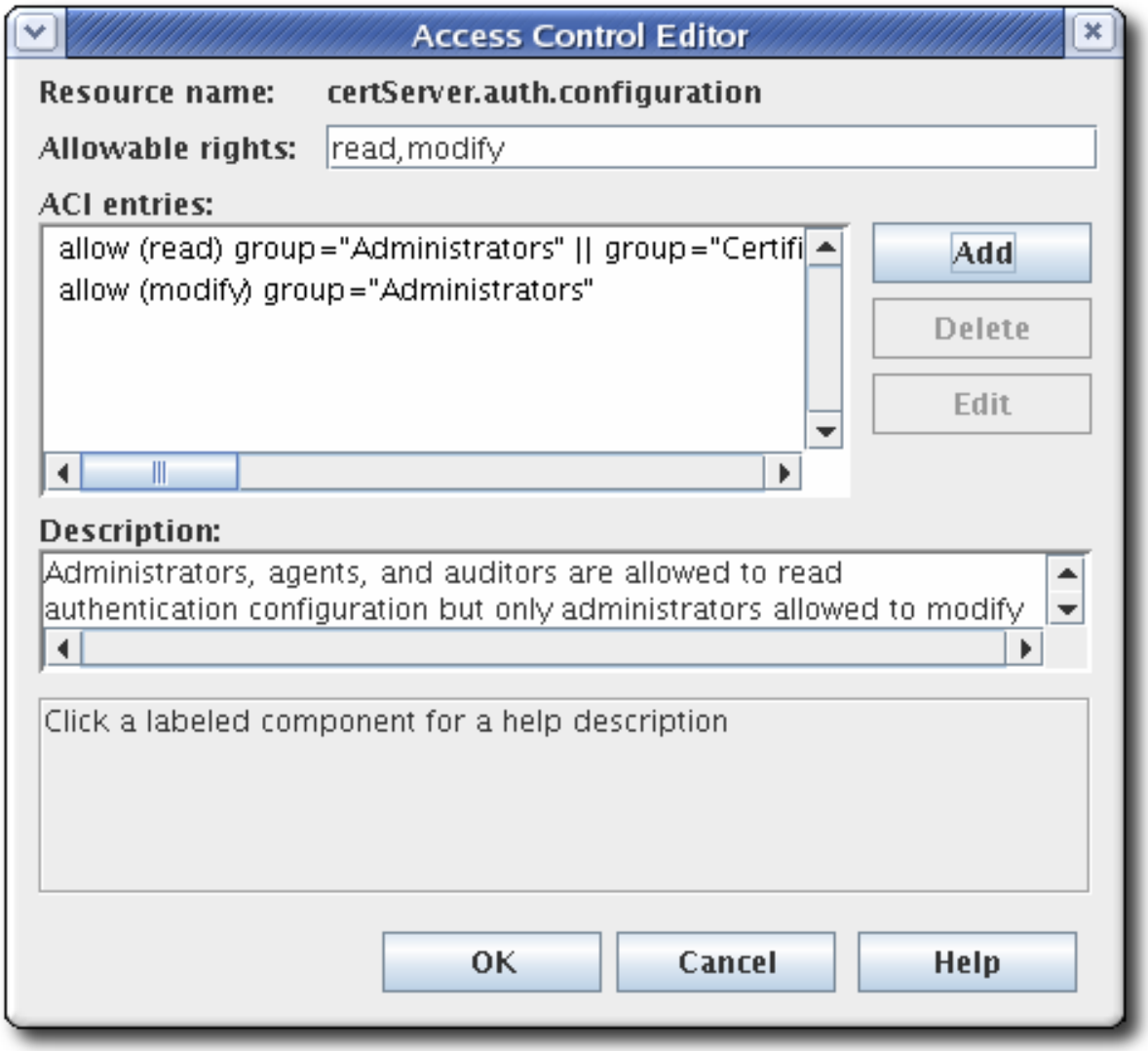
1. *登录到管理控制台。*

2. *在左侧导航菜单中选择 Access Control List。*



3. *从列表中选择要编辑的 ACL，然后单击 编辑。*

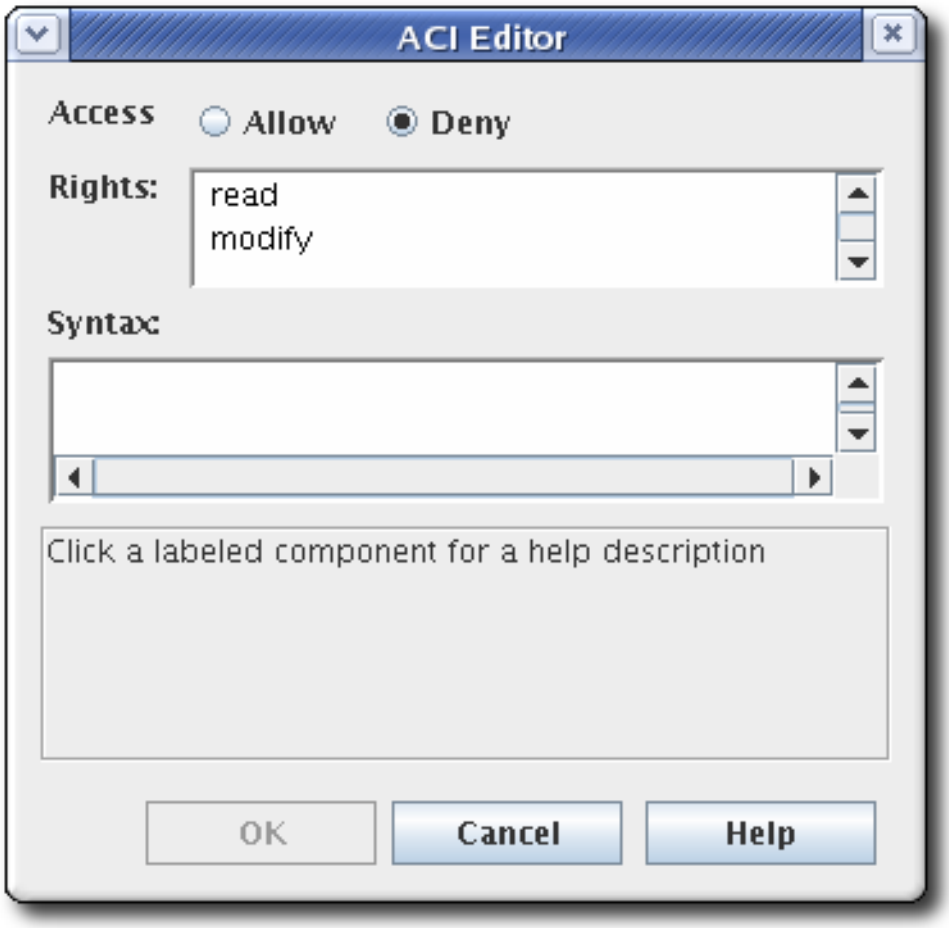
ACL 在 Access Control Editor 窗口中打开。



4.

***要添加 ACI，** 请单击 **Add** 并提供 **ACI** 信息。*

要编辑 ACI，** 请从 **ACL Editor** 窗口的 **ACI** 条目 文本区域中的列表中选择 **ACI。** 点 **Edit。



a.

从 Access 字段中选择允许或拒绝单选按钮，以允许或拒绝指定的组、用户或 IP 地址。有关允许或拒绝访问的详情，请参考 第 14.5.1 节 “关于访问控制”。

b.

为访问控制设置权限。这些选项是 读取 和修改。要设置两者，请使用 Ctrl 或 Shift 按钮。

c.

指定在 syntax 字段中授予或拒绝访问的用户、组或 IP 地址。有关语法的详情，请查看 第 14.5.1 节 “关于访问控制”。

第 15 章 配置子系统日志

CertificateCertificate Systemnbsp;System 子系统创建记录与活动相关的事件的日志文件，如管理、使用服务器支持的任何协议进行通信，以及由子系统使用的各种其他进程。在子系统实例运行时，它会保留有关它管理的所有组件的信息和错误消息。另外，Apache 和 Tomcat web 服务器也会生成错误并访问日志。

每个子系统实例维护自己的日志文件用于安装、审计和其他日志记录功能。

log 插件模块是作为 Java™ 类实施的监听程序，并在配置框架中注册。

除审计日志外，所有日志文件和轮转日志文件都位于通过 pkispawn 创建实例时在 pki_subsystem_log_path 中指定的任何目录中。 常规审计日志位于日志目录中，而签名的审计日志被写入 /var/log/pki/instance_name/subsystem_name/signedAudit。可以通过修改配置来更改日志的默认位置。

15.1. 关于证书证书系统NBSP;系统日志

CertificateCertificate Systemnbsp;System 子系统保留几种不同类型的日志，根据子系统类型、服务和单个日志设置提供特定信息。可以为实例保留的日志的类型取决于其所在的子系统。

15.1.1. 系统日志

子系统日志保留用于 CA、OCSP、KRA 和 TKS 子系统。

此日志、系统 记录了对服务器（所有 HTTP 和 HTTPS 请求）的请求的信息以及来自服务器的响应。此日志中记录的信息包括访问服务器的客户端机器的 IP 地址（包括 IPv4 和 IPv6）、执行的操作（如搜索、添加和编辑）；以及访问的结果，如返回的条目数量：

```
id_number processor - [date:time] [number_of_operations] [result] servlet: message
```

例 15.1. TKS 系统日志

```
10439.http-13443-Processor25 - [19/May/2020:14:16:51 CDT] [11] [3] UGSubsystem: Get User Error User not found
```

此日志默认是 on。

15.1.2. 事务日志

事务日志保留用于 CA、OCSP、KRA 和 TKS 子系统。

此日志、事务、记录任何执行或提交至子系统的操作。

```
id_number.processor - [date:time] [number_of_operations] [result] servlet: message
```

这些消息特定于证书服务，如 CA 接收证书请求、KRA 归档或检索密钥以及 TKS 注册新的 TPS。此日志也可用于检测任何未授权的访问或活动。

例 15.2. 事务日志

```
11438.http-8443-Processor25 - [27/May/2020:07:56:18 CDT] [1] [1] archival reqID 4
fromAgent agentID: CA-server.example.com-8443 authenticated by noAuthManager is
completed DN requested: UID=recoverykey,E=recoverykey@email.com,CN=recover key
serial number: 0x3
```

此日志默认是 on。

15.1.3. 调试日志

默认启用的调试日志为所有子系统维护，具有不同程度和类型的信息。

每个子系统的调试日志记录比系统、事务和访问日志更详细的信息。调试日志包含子系统执行的每个操作的特定信息，包括运行的插件和 servlet，以及服务器请求和响应消息。

第 15.2.1.1 节 “Are Logged 的服务” 中会简要讨论记录在 debug 日志中的常规服务类型。这些服务包括授权请求、处理证书请求、证书状态检查以及存档和恢复密钥以及访问 Web 服务。

CA、OCSP、KRA 和 TKS 记录与子系统进程相关的信息。每个日志条目的格式如下：

```
[date:time] [processor]: servlet: message
```

消息 可以是来自子系统的返回消息，或者包含提交到子系统的值。

例如，TKS 记录这个信息以连接到 LDAP 服务器：

```
[10/Jun/2020:05:14:51][main]: Established LDAP connection using basic authentication to
host localhost port 389 as cn=Directory Manager
```

处理器 是 主，而 消息 是来自有关 LDAP 连接的服务器的消息，没有 servlet。

另一方面，CA 记录有关证书操作以及子系统连接的信息：

```
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestowner$ value=KRA-server.example.com-8443
```

在这种情况下，处理器 是通过 CA 的代理端口的 HTTP 协议，而它指定了用于处理配置集的 servlet，并且 包含一条 提供配置文件参数（请求的子系统所有者）及其值（KRA 启动请求）。

例 15.3. CA 证书请求日志消息

```
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.profileapprovedby$ value=admin
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.cert_request$
value=MIIBozCCAZ8wggEFAgQqTfoHMIHHgAECpQ4wDDEKMAgGA1UEAxMBeKaBnzANB
gkqhkiG9w0BAQEFAAOB...
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.profile$ value=true
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.cert_request_type$ value=crmf
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestversion$ value=1.0.0
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.req_locale$ value=en
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestowner$ value=KRA-server.example.com-8443
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.dbstatus$ value=NOT_UPDATED
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.subject$ value=uid=jsmith, e=jsmith@example.com
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requeststatus$ value=begin
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.user$ value=uid=KRA-server.example.com-
8443,ou=People,dc=example,dc=com
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.req_key$
value=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxv
kLP^M
HcN0cusY7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChV^M
```

```
k9HYDhmJ8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5l+2gE9PUznxJaM^M
HTmlOqm4HwFxzy0RRQIDAQAB
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.authmgrinstname$ value=raCertAuth
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.uid$ value=KRA-server.example.com-8443
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.userid$ value=KRA-server.example.com-8443
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestor_name$ value=
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.profileid$ value=caUserCert
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.userdn$ value=uid=KRA-server.example.com-
4747,ou=People,dc=example,dc=com
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestid$ value=20
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.authtime$ value=1212782378071
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.req_x509info$
value=MIICIKADAgECAgEAMA0GCSqGSIb3DQEBBQUAMEAxHjAcBgNVBAoTFVJlZGJ1ZG
Nv^M
bXB1dGVyIERvbWFpbjEeMBwGA1UEAxMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4X^M
DTA4MDYwNjE5NTkzOFoXDTA4MTIwMzE5NTkzOFowOzEhMB8GCSqGSIb3DQEJARYS^M
anNtaXRoQGV4YW1wbGUuY29tMRYwFAYKCZImiZPyLGQBARMGanNtaXRoMIGfMA0G^M
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxvkLPHcN0cusY^M
7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChVk9HYDhmJ^M
8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5l+2gE9PUznxJaMHTmlOqm4^M
HwFxzy0RRQIDAQABo4HFMIHCMB8GA1UdIwQYMBaAFG8gWeOJlMt+aO8VuQTMzPBU^M
78k8MEoGCCsGAQUFBwEBBD4wPDA6BggrBgEFBQcwAYYuaHR0cDovL3Rlc3Q0LnJl^M
ZGJ1ZGNvbXB1dGVyLmxvY2FsOjkwODAvY2Evb2NzcDAOBgNVHQ8BAf8EBAMCBeAw^M
HQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMCQGA1UdEQQdMBuBGSRyZXF
1^M
ZXN0LnJlcXVlc3RvcI9lbWFpbCQ=
```

同样，OCSP 显示 OCSP 请求信息：

```
[07/Jul/2020:06:25:40][http-11180-Processor25]: OCSPServlet: OCSP Request:
[07/Jul/2020:06:25:40][http-11180-Processor25]: OCSPServlet:
MEUwQwIBADA+MDwwOjAJBgUrDgMCGgUABBSEWjCarLE6/BiSiENSsV9kHjqB3QQU
```

15.1.3.1. 安装日志

所有子系统均保留安装日志。

每次通过初始安装创建子系统时，或使用 pkispawn 创建额外实例，这是安装中的完整调试输出的安装文件，包括任何错误，包括任何错误，如果安装成功，则 URL 和 PIN 到实例的配置接口。该文件在

/var/log/pki/ 目录中创建，其名称为 pki-subsystem_name-spawn.timestamp 。

安装日志中的每个行都遵循安装过程的一个步骤。

例 15.4. CA Install Log

```
...
2015-07-22 20:43:13 pkispawn    : INFO     ... finalizing
'pki.server.deployment.scriptlets.finalization'
2015-07-22 20:43:13 pkispawn    : INFO     ..... cp -p /etc/sysconfig/pki/tomcat/pki-
tomcat/ca/deployment.cfg /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chmod 660 /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chown 26445:26445 /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : INFO     ..... generating manifest file called
'/etc/sysconfig/pki/tomcat/pki-tomcat/ca/manifest'
2015-07-22 20:43:13 pkispawn    : INFO     ..... cp -p /etc/sysconfig/pki/tomcat/pki-
tomcat/ca/manifest /var/log/pki/pki-tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chmod 660 /var/log/pki/pki-
tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chown 26445:26445 /var/log/pki/pki-
tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl enable pki-
tomcatd.target'
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl daemon-reload'
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl restart pki-
tomcatd@pki-tomcat.service'
2015-07-22 20:43:14 pkispawn    : INFO     END spawning subsystem 'CA' of instance 'pki-
tomcat'
2015-07-22 20:43:14 pkispawn    : DEBUG
```

15.1.3.2. Tomcat 错误和访问日志

CA、KRA、OCSP、TKS 和 TPS 子系统将 Tomcat Web 服务器实例用于代理和最终用户接口。

访问日志由 Tomcat web 服务器创建，该服务器随 CertificateCertificate Systemnbsp 一起安装；系统并提供 HTTP 服务。错误日志包含服务器遇到的 HTTP 错误消息。访问日志通过 HTTP 接口列出访问活动。

Tomcat 创建的日志：

- *admin.timestamp*
- *catalina.timestamp*
- *catalina.out*
- *host-manager.timestamp*
- *localhost.timestamp*
- *localhost_access_log.timestamp*
- *manager.timestamp*

这些日志在 CertificateCertificate Systemnbsp;System; 中不可配置，它们只能在 Apache 或 Tomcat 中进行配置。有关配置这些日志的详情，请查看 Apache 文档。

15.1.3.3. self-Tests 日志

当服务器启动时，或者在服务器启动时或自签名证书运行时，在 self-tests 运行过程中获取的 self-tests 记录信息。可以通过打开此日志来查看测试。此日志无法通过控制台配置，只能通过更改 CS.cfg 文件中的设置来配置。有关如何通过编辑 CS.cfg 文件配置日志的说明，请参阅 红帽证书系统规划、安装和部署指南中的 启用 Publishing Queue 部分。

本节中有关日志的信息与这个日志无关。有关自我证明的更多信息，请参阅 第 13.9 节 “运行自助测试”。

15.2. 管理日志

CertificateCertificate Systemnbsp;System 子系统日志文件记录与该特定子系统实例相关的事件。对于每个子系统，会保留不同的日志来用于安装、访问和 Web 服务器等问题。

所有子系统都有类似的日志配置、选项和管理路径。

15.2.1. 日志设置概述

配置日志的方式可能会影响 CertificateCertificate Systemnbsp;System performance。例如，日志文件轮转日志不会变得太大，这会降低子系统性能。本节介绍由 CertificateCertificate Systemnbsp 记录的不同日志；系统子系统，并涵盖日志文件轮转、缓冲日志记录以及可用日志级别等重要概念。

15.2.1.1. Are Logged 的服务

CertificateCertificate Systemnbsp 的所有主要组件和协议;系统日志消息到日志文件。表 15.1 “服务日志” 列出默认记录的服务。要查看特定服务记录的消息，请相应地自定义日志消息。详情请查看第 15.3.1 节 “在控制台中查看日志”。

表 15.1. 服务日志

Service	描述
ACL	日志与访问控制列表相关的事件。
管理	记录与管理活动相关的事件，如控制台和实例之间的 HTTPS 通信。
All	记录与所有服务相关的事件。
身份验证	使用身份验证模块记录与活动相关的事件。
证书颁发机构	记录与证书管理器相关的事件。
数据库	记录与内部数据库活动相关的事件。
HTTP	记录与服务器 HTTP 活动相关的事件。请注意，HTTP 事件实际上被记录到属于带有 CertificateCertificate Systemnbsp 的 Apache 服务器出错日志；System to 提供 HTTP 服务。
密钥恢复授权	记录与 KRA 相关的事件。
LDAP	记录与 LDAP 目录相关的事件，用于发布证书和 CRL。

Service	描述
OCSP	记录与 OCSP 相关的事件，如 OCSP 状态 GET 请求。
其他	记录与其他活动相关的事件，如命令行实用程序和其他进程。
请求队列	记录与请求队列活动相关的事件。
用户和组群	记录与实例的用户和组相关的事件。

15.2.1.2. 日志级别(Message Categories)

由 CertificateCertificate Systemnbsp 记录的不同事件; 系统服务由日志级别决定，这使得识别和过滤事件更简单。不同的证书证书系统nbsp;System 日志级别列在 表 15.2 “日志级别和更正日志消息” 中。

日志级别按数字表示，指示服务器应如何执行日志记录级别。

更高的优先级级别意味着不详细，因为只记录高优先级的事件。

表 15.2. 日志级别和更正日志消息

日志级别	消息类别	描述
0-1	Tracing	这些消息包含精细的调试信息。此级别不应定期使用，因为它可能会影响性能。
2-5	调试	这些消息包含调试信息。不建议使用这个级别，因为它会生成太多的信息。
6-10	INFORMATIONAL	这些消息提供有关证书证书证书系统状态的一般信息；System，包括证书证书系统nbsp 等状态消息 ；System initialization complete and Request for operation successful。
11-15	警告	这些消息只是警告信息，且不会指示服务器正常操作中的任何故障。

日志级别	消息类别	描述
> 15	失败	这些消息表示导致服务器正常运行的错误和失败，包括执行证书服务操作失败（用户身份 验证失败 或证书被撤销的）和意外情况，可能导致不相关的错误（服务器无法通过从客户端发出的同一频道来发回请求）。设置以上 15 级将尽量减少日志，因为只会记录失败。

日志级别可用于根据事件的严重性过滤日志条目。默认日志级别为 10。

日志数据可能广泛，特别是较低（详细）日志记录级别。确保主机机器具有足够的磁盘空间用于所有日志文件。还要适当定义日志级别、日志轮转和 server-backup 策略，以便备份所有日志文件，并且主机系统不会过载；否则，信息可能会丢失。

15.2.1.3. buffered 和 Unbuffered Logging

Java 子系统支持对所有类型的日志进行缓冲记录。可以为缓冲或未缓冲日志记录配置服务器。

如果配置了缓冲的日志，服务器会为相应日志创建缓冲区，并在缓冲区中尽可能保存消息。只有在发生以下条件之一时，服务器才会将信息清除到日志文件中：

- 缓冲区已满。当缓冲区大小等于或大于由 bufferSize 配置参数指定的值时，缓冲已满。此参数的默认值为 512 KB。
- 达到缓冲区的冲刷间隔。当从最后一个缓冲区刷新开始的时间间隔等于或大于 flushInterval 配置参数指定的值时，就会达到冲刷间隔。此参数的默认值为 5 秒。
- 当当前日志从控制台读取时。查询当前日志时，服务器会检索最新的日志。

如果为非缓冲记录配置了服务器，服务器会在生成日志时清除消息。由于服务器在每次生成消息时都执行 I/O 操作（写入日志文件），因此配置服务器以获得不缓冲的日志会降低性能。

设置日志参数在 第 15.2.2 节 “在控制台中配置日志” 中描述。

15.2.1.4. 日志文件轮转

子系统日志具有可选日志设置，允许轮转并启动新的日志文件，而不必让日志文件无限期地增加。在以下情况下之一会轮转日志文件：

- 达到对应文件的大小限制。对应日志文件的大小等于或大于 maxFileSize 配置参数指定的值。此参数的默认值为 100 KB。
- 相应文件的 age 限值会被达到。对应的日志文件等于 rolloverInterval 配置参数指定的时间间隔，或早于 rolloverInterval 配置参数。此参数的默认值为 2592000 秒（每隔三十天）。

注意

将这两个参数都设置为 0 可有效地禁用日志文件轮转。

轮转日志文件时，会使用带有附加时间戳的文件的名称命名旧文件。附加的时间戳是一个整数，表示相应活跃日志文件轮转的日期和时间。日期和时间的格式为 YYYYMMDD（年、月份、日）和 HHMMSS（小时、分钟、秒）。

日志文件（特别是审计日志文件）包含关键信息。通过将整个 日志 目录复制到归档中，应定期将这些文件归档到某些备份介质。

注意

CertificateCertificate Systemnbsp;System 不会为归档日志文件提供任何工具或实用程序。

CertificateCertificate Systemnbsp;System 提供了一个命令行实用程序, signtool，该实用程序在存档日志文件作为 tamper 检测方法之前对其进行签名。详情请查看 第 15.2.4.5 节 “签名日志文件”。

签名日志文件是签名的审计日志功能的替代选择。签名的审计日志会创建使用子系统签名证书的审计日志。有关签名的审计日志的详情，请参阅 第 15.2.4.3 节 “在控制台中配置签名审计日志”。

轮转的日志文件不会被删除。

15.2.2. 在控制台中配置日志

可以通过子系统控制台和子系统的 CS.cfg 文件配置日志。也可以通过控制台或配置文件创建专用日志，如签名的审计日志和自定义日志。

可使用 CA、OCSP、TKS 和 KRA 子系统的子系统控制台配置系统、事务和审计日志。TPS 日志仅通过配置文件进行配置。

1. 在 Configuration 选项卡的导航树中，选择 Log。

2. Log Event Listener Management 选项卡列出当前配置的监听程序。

   要创建新日志实例，请单击 Add，然后从 Select Log Event Listener Plug-in Implementation 窗口中的列表选择一个模块插件。

3. 设置或修改 Log Event Listener Editor 窗口中的字段。表 15.3 “日志记录事件 Listener 字段” 中列出了不同的参数。

表 15.3. 日志记录事件 Listener 字段

字段	描述
日志事件 Listener ID	赋予标识监听器的唯一名称。名称可具有任意字母（A 到 zZ）、数字（0 到 9）、一个下划线(_)和连字符(-)，但它不能包含其他字符或空格。
type	给出日志文件的类型。系统 会创建错误和系统日志；事务 记录审计日志。
enabled	设定日志是否活跃。仅启用的日志可实际记录事件。该值可以是 true 或 false。
level	在文本字段中设置日志级别。级别必须在字段中手动输入；没有选择菜单。选择是 Debug、信息、Warning、Failure、Misconfiguration、Catastrophe 和 Security。更多信息请参阅 第 15.2.1.2 节 “日志级别(Message Categories)”。
fileName	会向日志文件指定完整路径，包括文件名。子系统用户应具有对该文件的读/写入权限。

字段	描述
bufferSize	以 KB(KB)为单位设置日志的缓冲区大小。当缓冲区达到这个大小后，缓冲区的内容会被清除并复制到日志文件中。默认大小为 512 KB。有关缓冲日志记录的更多信息，请参阅 第 15.2.1.3 节 “buffered 和 Unbuffered Logging”。
flushInterval	设置在缓冲区内容清除并添加到日志文件中的时间。默认间隔为 5 秒。
maxFileSize	以 KB(KB)为单位设置大小，日志文件可以在轮转之前变为。达到这个大小后，文件会被复制到轮转的文件中，日志文件将启动新的。有关日志文件轮转的更多信息，请参阅 第 15.2.1.4 节 “日志文件轮转”。默认大小为 2000 KB。
rolloverInterval	设置服务器的频率来轮转活跃日志文件。可用选项包括 hourly、daily、weekly、monthly 和 year。默认是 monthly。更多信息请参阅 第 15.2.1.4 节 “日志文件轮转”。

15.2.3. 在 CS.cfg 文件中配置日志

有关如何通过编辑 CS.cfg 文件配置日志的说明，请参考 Red Hat 证书系统规划、安装和部署指南中的 CS.cfg 文件配置日志 部分。

15.2.4. 管理审计日志

审计日志包含已设置为可记录事件的事件的记录。如果 logSigning 属性设为 true，则审计日志使用属于服务器的日志签名证书签名。审核员可使用此证书来验证日志没有被篡改。

默认情况下，常规审计日志位于 /var/log/pki/instance_name/subsystem_name/ 目录，其他类型会写入 /var/log/pki/instance_name/ subsystem_name/ signedAudit/。可以通过修改配置来更改日志的默认位置。

签名的审计日志会创建一个记录系统事件的日志，事件是从潜在的事件列表中选择的事件。启用后，经过签名的审计日志记录一组有关所选事件活动的信息。

默认情况下，签名的审计日志是在实例首次创建时配置，但可以在安装后配置已签名的审计日志。（请参阅 第 15.2.4.2 节 “安装后启用签名审计日志”。） 您还可以编辑配置或更改配置后的签名证书，如 第 15.2.4.3 节 “在控制台中配置签名审计日志” 中所述。

15.2.4.1. 审计事件列表

有关证书系统中的审计事件列表，请参阅 附录 E, 审计事件。

15.2.4.2. 安装后启用签名审计日志

当实例首次创建时，可以使用 pkispawn 命令的 pki_audit_group 部署参数启用签名的审计日志。但是，在创建实例时不会配置签名的审计日志，可以通过将审计日志目录的所有权重新分配给 auditor 系统用户组（如 pkiaudit ）来启用它们。

1. 停止实例：

```
systemctl stop pki-tomcatd@instance_name.service
```

2. 将签名的审计日志目录的组所有权设置为 PKI auditors 操作系统组，如 pkiaudit。这允许 PKI auditors 组中的用户具有对 signedAudit 目录的读取访问权限，以验证日志文件上的签名。无用户（除了 CertificateCertificate Systemnbsp;System user account, pkiuser）的用户应该有权访问此目录中的日志文件。

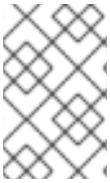
```
chgrp -R pkiaudit /var/log/pki/instance_name/subsystem_name/signedAudit
```

3. 重启实例：

```
systemctl start pki-tomcatd@instance_name.service
```

15.2.4.3. 在控制台中配置签名审计日志

默认情况下，签名的审计日志是在实例首次创建时配置，但可以在配置后编辑配置或更改签名证书。



注意

在文件系统中为签名的审计日志提供足够空间，因为它们大。

通过设置 logSigning 参数来启用并提供用于为日志签名的证书的别名，将日志设置为签名的审计日志。在首次配置子系统时，会创建一个特殊的日志签名证书。

只有具有审核员权限的用户才能访问和查看签名的审计日志。审核员可以使用 auditVerify 工具验证签名的审计日志是否已被篡改。

配置子系统时会创建并启用了签名的审计日志，但需要其他配置才能开始创建和签名审计日志。

1. 打开控制台。

注意

要通过编辑 CS.cfg 文件创建和配置审计日志，请参阅 红帽认证系统规划、安装和部署指南中的 CS.cfg 文件中的配置日志 部分。

2. 在 Configuration 选项卡的导航树中，选择 Log。

3. 在 Log Event Listener Management 选项卡中，选择 SignedAudit 条目。

4. 单击 编辑/查看。

5. 在 Log Event Listener Editor 窗口中必须重置三个字段。

- 填写 signedAuditCertNickname。这是用于为审计日志签名的证书的别名。在配置子系统时会创建审计签名证书；它有一个 nickname，如 auditSigningCert cert-instance_name 子系统_name。

注意

要获取审计签名证书 nickname，请使用 certutil 列出子系统的证书数据库。例如：

```
certutil -L -d /var/lib/pki-tomcat/alias

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-pki-tomcat           u,u,u
Server-Cert cert-pki-tomcat             u,u,u
auditSigningCert cert-pki-tomcat CA     u,u,Pu
```

- 将 logSigning 字段设置为 true 来启用签名日志记录。

- 设置日志记录到审计日志的任何事件。附录 E, 审计事件 列出可记录的事件。日志事件用逗号分开，且没有空格。

6. 为日志设置任何其他设置，如文件名、日志级别、文件大小或轮转调度。

注意

默认情况下，常规审计日志位于 /var/log/pki/instance_name/ 子系统_name /subsystem_name/ 目录中，而签名的审计日志被写入 /var/log/pki/instance_name /subsystem_name/ signedAudit/。可以通过修改配置来更改日志的默认位置。

7. 保存日志配置。

启用经过签名的审计日志记录后，通过创建用户并将该条目分配给 auditor 组来分配审核员用户。auditor 组的成员是唯一可以查看和验证签名的审计日志的用户。有关设置审核员的详情，请参阅 第 14.3.2.1 节 “创建用户”。

审核员可以使用 AuditVerify 工具验证日志。有关使用这个工具的详情，请查看 audit Verify(1) 手册页。

15.2.4.4. 处理审计日志故障

有一些事件可能会导致审计日志记录功能失败，因此事件无法写入日志。例如，当包含审计日志文件的文件系统已满或者日志文件的文件权限被意外改变时，审计日志记录可能会失败。如果审计日志记录失败，CertificateCertificate Systemnbsp;System instance 以下列方式关闭。

- Servlet 被禁用，不会处理新请求。

- 所有待定和新请求都会被终止。

- 该子系统已关机。

发生这种情况时，管理员和审核员应与操作系统管理员一起工作，以解决磁盘空间或文件权限问题。当 IT 问题解决时，审核员应确保对最后的审计日志条目进行签名。如果没有，则应该通过手动签名（第 15.2.4.5 节 “签名日志文件”）（归档和删除）来保留它们，以防止将来出现审核验证失败。此过程完成后，管理员可以重启证书证书系统nbsp;System。

15.2.4.5. 签名日志文件

CertificateCertificate Systemnbsp;System 可在存档或分发为审核前为日志文件进行数字签名。此功能允许检查文件以进行篡改。

这是签名的审计日志功能的替代选择。签名的审计日志功能会创建自动签名的审计日志；此工具手动签署存档日志。有关签名的审计日志的详情，请参阅 第 15.2.4.3 节 “在控制台中配置签名审计日志”。

要签名日志文件，请使用名为 Signing Tool 的命令行实用程序（签名工具）。有关这个工具的详情，请参考 http://www.mozilla.org/projects/security/pki/nss/tools/。

实用程序使用子系统实例的证书、密钥和安全模块数据库中的信息。

以具有 auditor 特权的用户身份，使用 signtool 命令签署日志目录：

```
signtool -d secdb_dir -k cert_nickname -Z output input
```

- secdb_dir 指定包含 CA 的证书、密钥和安全模块数据库的目录路径。
- cert_nickname 指定用于签名的证书的 nickname。
- 输出指定 JAR 文件的名称（签名的 zip 文件）。
- input 指定包含日志文件的目录的路径。

15.2.4.6. 过滤审计事件

在证书系统管理员中，可以设置过滤器来配置根据事件属性在审计文件中记录哪些审计事件。

过滤器的格式与 LDAP 过滤器的格式相同。但是，证书系统只支持以下过滤器：

表 15.4. 支持的审计事件过滤器

类型	格式	示例
存在	(attribute=*)	(ReqID=*)
等于	（属性=值）	(Outcome=Failure)
子字符串	(attribute=initial*any*...*any*final)	(SubjectID=*admin*)
AND 操作	(&(filter_1)(filter_2)...(filter_n))	(&(SubjectID=admin)(Outcome=Failure))
OR 操作	(|(filter_1)(filter_2)...(filter_n))	(|(SubjectID=admin)(Outcome=Failure))
不 操作	(!(filter))	(!(SubjectID=admin))

有关 LDAP 过滤器的详情，请参考 Red Hat Directory Server 管理指南 中的使用 复合 搜索过滤器。

例 15.5. 过滤审计事件

为处理处理的证书请求和事件记录失败的事件，并将 InfoName 字段设置为 rejectReadon 或 cancelReason ：

1. 编辑 /var/lib/pki/instance_name/subsystem_type/conf/CS.cfg 文件并设置以下参数：

```
log.instance.SignedAudit.filters.PROFILE_CERT_REQUEST=(Outcome=Failure)
log.instance.SignedAudit.filters.CERT_REQUEST_PROCESSED=(|
(InfoName=rejectReason)(InfoName=cancelReason))
```

2. 重启证书系统：

```
# systemctl restart pki-tomcatd@instance_name.service
```

-

15.2.5. 管理日志模块

允许的日志类型及其行为通过 log 模块 插件进行配置。可以创建新的日志记录模块，以用于设置自定义日志。

新的日志插件模块可以通过 Console 注册。注册新模块涉及指定模块的名称和实施日志接口的 Java™ 类的完整名称。

在注册插件模块之前，请将模块的 Java™ 类放在 类 目录中；实施必须在类路径上。

使用子系统实例注册日志插件模块：

1. 创建自定义作业类。在本例中，自定义日志插件名为 MyLog.java。

2. 将新类编译到实例的 lib 目录中。

```
javac -d . /var/lib/pki/pki-tomcat/lib -classpath $CLASSPATH MyLog.java
```

3. 在 CA 的 WEB-INF Web 目录中创建用于存放自定义类的目录，以便 CA 可以访问它们。

```
mkdir /var/lib/pki/pki-tomcat/webapps/ca/WEB-INF/classes
```

4. 将所有者设置为 CertificateCertificate Systemnbsp;System system user(pkiuser)。

```
chown -R pkiuser:pkiuser /var/lib/pki/pki-tomcat/lib
```

5. 注册插件。

   a. 登录到控制台。

   b. 在 Configuration 选项卡中，从导航树中选择 Logs。然后选择 Log Event Listener Plug-in Registration 选项卡。

c.
点 Register。

此时会出现 Register Log Event Listener Plug-in Implementation 窗口。

d.
为插件模块和 Java™ 类名称命名。

Java™ 类名称是实施 Java™ 类的完整路径。如果这个类是软件包的一部分，请包含软件包名称。例如，在名为 com.customplugins 的软件包中注册一个名为 customLog 的类，类名称为 com.customplugins.customLog。

e.
点击 OK。

不需要的 log 插件模块可以通过 Console 删除。在删除模块前，删除基于这个模块的所有监听程序；请参阅 第 15.2.1.4 节 “日志文件轮转”。

15.3. 使用日志

15.3.1. 在控制台中查看日志

要对子系统进行故障排除，请检查服务器已记录的错误或信息消息。检查日志文件也可以监控服务器操作的很多方面。可以通过控制台查看一些日志文件。但是，审计日志只能由具有 Auditor 角色的用户使用 第 15.3.2 节 “使用签名审计日志” 中详述的方法访问。

查看活跃或轮转的系统日志文件的内容：

1.
登录到控制台。

2.
选择 Status 选项卡。

3.
在 Logs 下，选择要查看的日志。

4.
在 Display Options 部分中设置查看首选项。

- entries - 要显示的最大条目数。达到这个限制时，CertificateCertificate Systemnbsp;System 会返回与搜索请求匹配的条目。零(0)表示没有返回任何信息。如果字段为空，服务器会返回每个匹配的条目，无论找到的数量如何。

- Source - 选择证书Certificate Systemnbsp; 系统组件或服务，用于显示哪些日志消息。选择 All 表示日志记录到此文件的所有组件的消息。

- level - 选择代表过滤消息的日志级别的消息类别。

- 文件名 - 选择要查看的日志文件。选择 Current 来查看当前活跃的系统日志文件。

5. 点 Refresh。

下表显示了系统日志条目。条目以反向时间顺序排列，当前条目位于顶部。使用面板右边的滚动箭头滚动浏览日志条目。

每个条目都显示以下信息：

- Source - 记录消息的组件或资源。

- level - 对应条目的严重性。

- date - 记录条目的日期。

- time - 记录条目的时间。

- details - 日志的简单描述。

6. 要查看完整条目，请双击该条目，或者选择 条目，然后单击 View。

15.3.2. 使用签名审计日志

本节介绍 Auditor 组中的用户如何显示和验证已签名的审计日志。

15.3.2.1. 列出审计日志

以具有 auditor 权限的用户身份，使用 pki 子系统-audit-file-find 命令列出服务器上的现有审计日志文件。

例如，列出 server.example.com 上托管的 CA 上的审计日志文件：

```
# pki -h server.example.com -p 8443 -n auditor ca-audit-file-find
----------------
3 entries matched
----------------
 File name: ca_audit.20170331225716
 Size: 2883

 File name: ca_audit.20170401001030
 Size: 189

 File name: ca_audit
 Size: 6705
----------------------------
Number of entries returned 3
----------------------------
```

该命令使用客户端证书，使 auditor nickname 存储在 ~/.dogtag/nssdb/ 目录中以向 CA 进行身份验证。有关命令和替代验证方法中使用的参数的详情，请查看 pki(1) man page。

15.3.2.2. 下载审计日志

以具有 auditor 特权的用户身份，使用 pki subsystem-audit-file-retrieve 命令从服务器下载特定的审计日志。

例如，要从 server.example.com 上托管的 CA 下载审计日志文件：

1. （可选）列出 CA 上可用的日志文件。请参阅 第 15.3.2.1 节 “列出审计日志”。

2. 下载日志文件。例如，要下载 ca_audit 文件：

```
# pki -U https://server.example.com:8443 -n auditor ca-audit-file-retrieve ca_audit
```

该命令使用客户端证书，使 auditor nickname 存储在 ~/.dogtag/nssdb/ 目录中以向 CA 进行身份验证。有关命令和替代验证方法中使用的参数的详情，请查看 pki(1) man page。

下载日志文件后，您可以使用 grep 实用程序搜索特定的日志条目，例如：

```
# grep "\[AuditEvent=ACCESS_SESSION_ESTABLISH\]" log_file
```

15.3.2.3. 验证签名审计日志

如果启用了审计日志签名，则具有审核员权限的用户可以验证日志：

1. 初始化 NSS 数据库并导入 CA 证书。详情请参阅 Red Hat Certificate System 9 规划、安装和部署指南中的 第 2.5.1.1 节 “pki CLI initialization” 和 将 证书导入到 NSS 数据库 部分。

2. 如果 PKI 客户端数据库中不存在审计签名证书，请导入：

   a. 搜索您要验证的子系统日志的审计签名证书。例如：

   ```
   # pki ca-cert-find --name "CA Audit Signing Certificate"
   ---------------
   1 entries found
   ---------------
    Serial Number: 0x5
    Subject DN: CN=CA Audit Signing Certificate,O=EXAMPLE
    Status: VALID
    Type: X.509 version 3
    Key Algorithm: PKCS #1 RSA with 2048-bit key
    Not Valid Before: Fri Jul 08 03:56:08 CEST 2016
    Not Valid After: Thu Jun 28 03:56:08 CEST 2018
    Issued On: Fri Jul 08 03:56:08 CEST 2016
    Issued By: system
   ----------------------------
   Number of entries returned 1
   ----------------------------
   ```

   b. 将审计签名证书导入到 PKI 客户端中：

   ```
   # pki client-cert-import "CA Audit Signing Certificate" --serial 0x5 --trust ",,P"
   ```

```
---------------------------------------------------
Imported certificate "CA Audit Signing Certificate"
---------------------------------------------------
```

3. 下载审计日志。请参阅 第 15.3.2.2 节 “下载审计日志”。

4. 验证审计日志。

   a. 创建一个文本文件，其中包含您要按时间顺序验证的审计日志文件列表。例如：

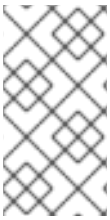
```
# cat > ~/audit.txt << EOF
ca_audit.20170331225716
ca_audit.20170401001030
ca_audit
EOF
```

   b. 使用 AuditVerify 程序验证签名。例如：

```
# AuditVerify -d ~/.dogtag/nssdb/ -n "CA Audit Signing Certificate" \
   -a ~/audit.txt
Verification process complete.
Valid signatures: 10
Invalid signatures: 0
```

   有关使用 AuditVerify 的详情，请查看 AuditVerify(1) man page。

15.3.3. 显示操作系统级别审计日志



注意

要查看以下说明的操作系统级审计日志，必须根据 红帽认证系统 9 规划、安装和部署指南中的 启用操作系统级别的 Audit 日志 部分配置 auditd 日志。

要显示操作系统级别的访问日志，请使用 ausearch 工具程序作为 root 用户，或使用 sudo 程序作为特权用户。

15.3.3.1. 显示审计日志删除事件

由于这些事件被密钥（使用 rhcs_audit_deletion），因此请使用 -k 参数查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_deletion
```

15.3.3.2. 显示对 Secret 和私钥的 NSS 数据库的访问

由于这些事件被密钥（使用 rhcs_audit_nssdb），因此请使用 -k 参数查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_nssdb
```

15.3.3.3. 显示时间更改事件

由于这些事件被密钥（使用 rhcs_audit_time_change），因此请使用 -k 参数查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_time_change
```

15.3.3.4. 显示软件包更新事件

由于这些事件是键入的消息（类型为 SOFTWARE_UPDATE），因此请使用 -m 参数查找与该类型匹配的事件：

```
# ausearch -m SOFTWARE_UPDATE
```

15.3.3.5. 显示 PKI 配置的更改

由于这些事件被密钥（使用 rhcs_audit_config），因此请使用 -k 参数查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_config
```

15.3.4. 智能卡错误代码

智能卡可能会向 TPS 报告某些错误代码，这些代码记录在 TPS 的 debug 日志文件中，具体取决于消息的原因。

表 15.5. 智能卡错误代码

返回码	描述
常规错误代码	
6400	没有特定诊断
6700	Lc 中错误的长度
6982	未满足安全状态
6985	不满足使用的条件
6a86	P1 P2 错误
6d00	无效的指令
6e00	无效的类
安装 Load Errors	
6581	内存故障
6a80	data 字段中的参数不正确
6a84	没有足够的内存空间
6a88	未找到引用的数据
删除错误	
6200	应用程序已被逻辑删除
6581	内存故障
6985	不能删除引用的数据
6a88	未找到引用的数据
6a82	未找到应用程序
6a80	命令数据中的不正确的值

返回码	描述
获取数据错误	
6a88	未找到引用的数据
获取状态错误	
6310	可用数据更多
6a88	未找到引用的数据
6a80	命令数据中的不正确的值
加载错误	
6581	内存故障
6a84	没有足够的内存空间
6a86	P1/P2 错误
6985	不满足使用的条件

第 16 章 管理子系统证书

本章概述了使用证书：哪些类型和格式可用，如何通过 HTML 最终用户形式来请求和创建它们，并通过证书证书系统nbsp 并通过证书证书系统及证书控制台(System Console)在证书证书系统中安装证书，以及如何在不同客户端上安装证书。另外，还有通过控制台管理证书以及配置服务器以使用它们的信息。

16.1. 所需的 子系统证书

每个子系统都有一组定义的证书，必须发布到子系统实例，以便它执行操作。在证书管理器配置期间设置某些证书内容的详情，根据证书类型限制、设置和属性的不同考虑；规划证书格式包括在 Red Hat Certificate System 9 Planning、Installing and Deployment Guide 中。

16.1.1. 证书管理器证书

安装证书管理器时，将生成 CA 签名证书、SSL 服务器证书和 OCSP 签名证书的密钥和请求。证书会在配置完成前创建。

CA 证书请求可作为自签名请求提交给 CA，然后发布证书并完成创建自签名 root CA，或提交至第三方公共 CA 或其他证书证书系统nbsp;System CA。当外部 CA 返回证书时，会安装证书并安装从属 CA。

- 第 16.1.1.1 节 “CA 签名密钥对和证书”
- 第 16.1.1.2 节 “OCSP 签名密钥对和证书”
- 第 16.1.1.3 节 “子系统证书”
- 第 16.1.1.4 节 “SSL 服务器密钥对和证书”
- 第 16.1.1.5 节 “审计日志签名密钥对和证书”

16.1.1.1. CA 签名密钥对和证书

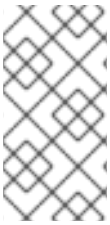
每个证书管理器都有一个 CA 签名证书，其公钥对应于证书管理器用来为证书和 CRL 证书签名。安装证书管理器时会创建并安装此证书。证书的默认 nickname 是 caSigningCert-instance_ID CA，其中

instance_ID 标识证书管理器实例。证书的默认有效期期是五年。

CA 签名证书的主题名称反映了在安装过程中设置的 CA 名称。证书管理器签名或发布的所有证书均包括此名称来识别证书的签发者。

证书管理器的状态是作为 root 或从属 CA 的状态，由其 CA 签名证书是自签名还是由另一个 CA 签名，后者会影响证书上的主题名称。

- 如果证书管理器是 root CA，则其 CA 签名证书是自签名的，即证书的主题名称和签发者名称相同。
- 如果证书管理器是从属 CA，则其 CA 签名证书由另一个 CA 签名，通常是位于 CA 层次结构的级别（可能或不能是 root CA）中的级别。root CA 的签名证书必须导入到单独的客户端和服务器中，然后才能使用证书管理器向它们发布证书。



注意

无法更改 CA 名称，或者之前发布的所有证书都无效。同样，使用新密钥对退出 CA 签名证书，使所有由旧密钥对签名的证书无效。

16.1.1.2. OCSP 签名密钥对和证书

OCSP 签名证书的主题名称格式为 cn=OCSP cert-instance_ID CA，它包含扩展，如 OCSPSigning 和 OCSPNoCheck，在签名 OCSP 响应时需要。

OCSP 签名证书的默认 nickname 是 ocspSigningCert cert-instance_ID，其中 instance_ID CA 标识证书管理器实例。

OCSP 私钥（与 OCSP 签名证书的公钥）用于在查询证书撤销状态时，为与 OCSP 兼容的客户端签名。

16.1.1.3. 子系统证书

安全域的每个成员都发出一个服务器证书，用于其他域成员之间的通信，该证书与服务器 SSL 证书分开。此证书由安全域 CA 签名；对于安全域 CA，其子系统证书由自身签名。

证书的默认 nickname 是 subsystemCert cert-instance_ID。

16.1.1.4. SSL 服务器密钥对和证书

每个证书管理器至少包含一个 SSL 服务器证书，这是在安装证书管理器时首次生成的 SSL 服务器证书。证书的默认 nickname 是 Server-Cert cert-instance_ID，其中 instance_ID 标识证书管理器实例。

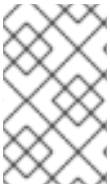
默认情况下，证书管理器使用单个 SSL 服务器证书进行身份验证。但是，可以为不同的操作请求额外的服务器证书，比如将证书管理器配置为使用单独的服务器证书来向最终用户接口和代理服务接口进行身份验证。

如果为启用了 SSL 的 SSL 通讯配置了证书管理器，则默认情况下，它使用其 SSL 服务器证书来向发布目录进行客户端身份验证。证书管理器也可以配置为使用其他证书进行 SSL 客户端身份验证。

16.1.1.5. 审计日志签名密钥对和证书

CA 保留服务器发生的所有事件的安全审计日志。为保证审计日志未被篡改，该日志文件由特殊日志签名证书签名。

第一次配置服务器时会发出审计日志签名证书。



注意

虽然其他证书可以使用 ECC 密钥，但审计签名证书 必须始终 使用 RSA 密钥。

16.1.2. 在线证书状态管理器证书

首次配置在线证书状态管理器时，会创建所有所需证书的密钥，以及对 OCSP 签名、SSL 服务器、审计日志签名和子系统证书的证书请求。这些证书请求将提交到 CA（证书证书系统侦听;System CA 或第三方 CA），且必须在在线证书状态管理器数据库中安装，以完成配置过程。

- 第 16.1.2.2 节 “SSL 服务器密钥对和证书”
- 第 16.1.2.3 节 “子系统证书”

- 第 16.1.2.4 节 “审计日志签名密钥对和证书”
- 第 16.1.2.5 节 “识别在线证书状态管理器证书”

16.1.2.1. OCSP 签名密钥对和证书

每个在线证书状态管理器都有一个证书，OCSP 签名证书，该证书具有与在线证书状态管理器用于为 OCSP 响应签名的私钥对应的公钥。Online Certificate Status Manager 的签名提供持久性证明，在线证书状态管理器已处理请求。当配置了在线证书 Status Manager 时，会生成此证书。证书的默认 nickname 是 ocspSigningCert cert-instance_ID，其中 instance_ID OSCP 是在线证书 Status Manager 实例名称。

16.1.2.2. SSL 服务器密钥对和证书

每个在线证书状态管理器都至少有一个 SSL 服务器证书，该证书在配置了在线证书状态管理器时生成。证书的默认 nickname 是 Server-Cert cert-instance_ID，其中 instance_ID 标识在线证书状态管理器实例名称。

Online Certificate Status Manager 使用其服务器证书进行在线证书状态代理服务页面的服务器端身份验证。

在线证书状态管理器使用单一服务器证书进行验证。可以安装其他服务器证书并将其用于不同的目的。

16.1.2.3. 子系统证书

安全域的每个成员都发出一个服务器证书，用于其他域成员之间的通信，该证书与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认 nickname 是 subsystemCert cert-instance_ID。

16.1.2.4. 审计日志签名密钥对和证书

OCSP 保留服务器发生的所有事件的安全审计日志。为保证审计日志未被篡改，该日志文件由特殊日志签名证书签名。

第一次配置服务器时会发出审计日志签名证书。

注意

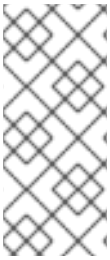
虽然其他证书可以使用 ECC 密钥，但审计签名证书 必须始终 使用 RSA 密钥。

16.1.2.5. 识别在线证书状态管理器证书

根据为在线证书状态管理器 SSL 服务器证书签名的 CA，可能需要获取证书并发布由证书管理器识别的 CA。

- 如果在线证书 Status Manager 的服务器证书由发布 CRL 的 CA 签名，则不需要进行任何操作。
- 如果在线证书状态管理器的服务器证书由签署从属证书管理器证书的同一 root CA 签名，那么 root CA 必须在从属证书管理器的证书数据库中被标记为可信 CA。
- 如果在线证书状态管理器的 SSL 服务器证书由不同的 root CA 签名，则必须将 root CA 证书导入到从属证书管理器的证书数据库中并标记为可信 CA。

如果在线证书状态管理器的服务器证书由所选安全域中的 CA 签名，则会导入证书链，并在配置在线证书 Status Manager 时标记。不需要其他配置。但是，如果服务器证书由外部 CA 签名，则必须导入证书链，以便完成配置。



注意

在配置了 OCSP Manager 时，不是安全域中的每个 CA 会自动被 OCSP Manager 信任。CA 面板中配置的 CA 证书链中的每个 CA 都是 OCSP Manager 自动信任的。安全域中的其他 CA，但必须在证书链中手动添加。

16.1.3. 密钥恢复授权证书

KRA 使用以下密钥对和证书：

- 第 16.1.3.1 节 “传输密钥对和证书”

- 第 16.1.3.2 节 “存储密钥对”
- 第 16.1.3.3 节 “SSL 服务器证书”
- 第 16.1.3.4 节 “子系统证书”
- 第 16.1.3.5 节 “审计日志签名密钥对和证书”

16.1.3.1. 传输密钥对和证书

每个 KRA 都有一个传输证书。用于生成传输证书的公钥，由客户端软件用于在向 KRA 发送至 KRA 进行归档之前加密最终用户的私钥；只有能够生成双密钥对的客户端使用传输证书。

16.1.3.2. 存储密钥对

每个 KRA 都有一个存储密钥对。 KRA 使用此密钥对的公钥在存档密钥时加密（或嵌套）私钥。它使用私有组件在恢复过程中解密（或取消包装）归档的密钥。有关使用此密钥对的更多信息，请参阅 第 4 章 设置密钥存档和恢复。

使用存储密钥加密的密钥只能被授权密钥恢复代理检索。

16.1.3.3. SSL 服务器证书

每个证书证书系统侦听;系统 KRA 至少有一个 SSL 服务器证书。配置 KRA 时会生成一个第一个 SSL 服务器证书。证书的默认 nickname 是 Server-Cert cert-instance_ID，其中 instance_id 标识 KRA 实例。

KRA 的 SSL 服务器证书由提交证书请求的 CA 发布，该证书可以是证书证书系统nbsp；System CA 或第三方 CA。要查看签发者名称，请在 KRA Console 中的 System Keys 和 Certificates 选项中打开证书详情。

KRA 使用其 SSL 服务器证书进行服务器端身份验证到 KRA 代理服务接口。默认情况下，密钥恢复授权使用单个 SSL 服务器证书进行验证。但是，可以为 KRA 请求并安装额外的 SSL 服务器证书。

16.1.3.4. 子系统证书

安全域的每个成员都发出一个服务器证书，用于其他域成员之间的通信，该证书与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认 nickname 是 subsystemCert cert-instance_ID。

16.1.3.5. 审计日志签名密钥对和证书

KRA 保留服务器发生的所有事件的安全审计日志。为保证审计日志未被篡改，该日志文件由特殊日志签名证书签名。

第一次配置服务器时会发出审计日志签名证书。

注意

虽然其他证书可以使用 ECC 密钥，但审计签名证书 必须始终 使用 RSA 密钥。

16.1.4. TKS 证书

TKS 有三个证书。SSL 服务器和子系统证书用于标准操作。额外的签名证书用于保护审计日志。

- 第 16.1.4.1 节 “SSL 服务器证书”
- 第 16.1.4.2 节 “子系统证书”
- 第 16.1.4.3 节 “审计日志签名密钥对和证书”

16.1.4.1. SSL 服务器证书

每个证书证书系统启动;系统 TKS 至少有一个 SSL 服务器证书。配置 TKS 时会生成第一个 SSL 服务器证书。证书的默认 nickname 是 Server-Cert cert-instance_ID。

16.1.4.2. 子系统证书

安全域的每个成员都发出一个服务器证书，用于其他域成员之间的通信，该证书与服务器 SSL 证书

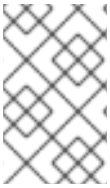
分开。此证书由安全域 CA 签名。

证书的默认 nickname 是 subsystemCert cert-instance_ID。

16.1.4.3. 审计日志签名密钥对和证书

TKS 保留服务器发生的所有事件的安全审计日志。为保证审计日志未被篡改，该日志文件由特殊日志签名证书签名。

第一次配置服务器时会发出审计日志签名证书。



注意

虽然其他证书可以使用 ECC 密钥，但审计签名证书 必须始终 使用 RSA 密钥。

16.1.5. TPS 证书

TPS 仅使用三个证书：服务器证书、子系统证书和审计日志签名证书。

- 第 16.1.5.1 节 “SSL 服务器证书”
- 第 16.1.5.2 节 “子系统证书”
- 第 16.1.5.3 节 “审计日志签名密钥对和证书”

16.1.5.1. SSL 服务器证书

每个证书证书系统nbsp;System TPS 至少一个 SSL 服务器证书。配置 TPS 时生成第一个 SSL 服务器证书。证书的默认 nickname 是 Server-Cert cert-instance_ID。

16.1.5.2. 子系统证书

安全域的每个成员都发出一个服务器证书，用于其他域成员之间的通信，该证书与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认 nickname 是 subsystemCert cert-instance_ID。

16.1.5.3. 审计日志签名密钥对和证书

TPS 保留服务器发生的所有事件的安全审计日志。为保证审计日志未被篡改，该日志文件由特殊日志签名证书签名。

第一次配置服务器时会发出审计日志签名证书。

16.1.6. 关于 subsystem 证书和密钥类型

在创建新实例时，您可以在传递到 pkispawn 实用程序的配置文件中指定密钥类型和密钥大小。

例 16.1. CA 的密钥类型相关配置参数

以下是类型与关键相关的参数，包括示例值。您可以在配置文件中设置这些参数，您可以在创建新 CA 时传递给 pkispawn。

```
pki_ocsp_signing_key_algorithm=SHA256withRSA
pki_ocsp_signing_key_size=2048
pki_ocsp_signing_key_type=rsa

pki_ca_signing_key_algorithm=SHA256withRSA
pki_ca_signing_key_size=2048
pki_ca_signing_key_type=rsa

pki_sslserver_key_algorithm=SHA256withRSA
pki_sslserver_key_size=2048
pki_sslserver_key_type=rsa

pki_subsystem_key_algorithm=SHA256withRSA
pki_subsystem_key_size=2048
pki_subsystem_key_type=rsa

pki_admin_keysize=2048
pki_admin_key_size=2048
pki_admin_key_type=rsa

pki_audit_signing_key_algorithm=SHA256withRSA
pki_audit_signing_key_size=2048
pki_audit_signing_key_type=rsa
```

注意

示例中的值适用于 CA。其他子系统需要不同的参数。

如需了解更多详细信息，请参阅：

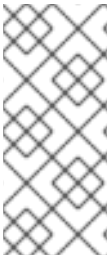
- 红帽证书系统规划、安装和部署指南中的 *了解 pkispawn 实用程序* 部分。
- pki_default.cfg(5) man page 了解参数和示例的描述。

16.1.7. 使用 HSM 存储子系统证书

默认情况下，密钥和证书保存在本地管理的数据库中，分别为 key3.db 和 cert8.db，在 /var/lib/pki/instance_name/alias 目录中。但是，Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System 还支持硬件安全模块(HSM)，可以在网络上集中存储密钥和证书的外部设备。使用 HSM 可以使一些功能（如克隆）变得容易，因为实例的密钥和证书可以被轻松访问。

当使用 HSM 存储证书时，HSM 名称会加上证书 nickname 的前面，并在子系统配置中使用完整名称，如 server.xml 文件。例如：

```
serverCert="nethsm:Server-Cert cert-instance_ID
```



注意

单个 HSM 可用于存储可在多个主机上安装的 mulitple 子系统实例的证书和密钥。使用 HSM 时，每个子系统的证书 nickname 都必须对 HSM 上管理的每个子系统实例是唯一的。

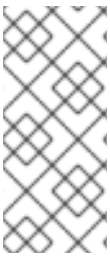
CertificateCertificate Systemnbsp;System 支持两种类型的 HSM，NCipher netHSM 和 Chrysalis LunaSA。

16.2. 通过控制台请求证书

CA、OCSP、KRA 和 TKS 的证书设置向导可以自动注册子系统证书的证书注册过程。控制台可以为该子系统使用的任何证书创建、提交和安装证书。这些证书可以是服务器证书或特定于子系统的证书，如

CA 签名证书或 KRA 传输证书。

16.2.1. 请求签名证书



注意

务必要从计算机生成并提交客户端请求，该请求稍后将用于访问子系统，因为请求过程的一部分在本地计算机上生成私钥。如果需要独立位置，请使用硬件令牌（如智能卡）存储密钥对和证书。

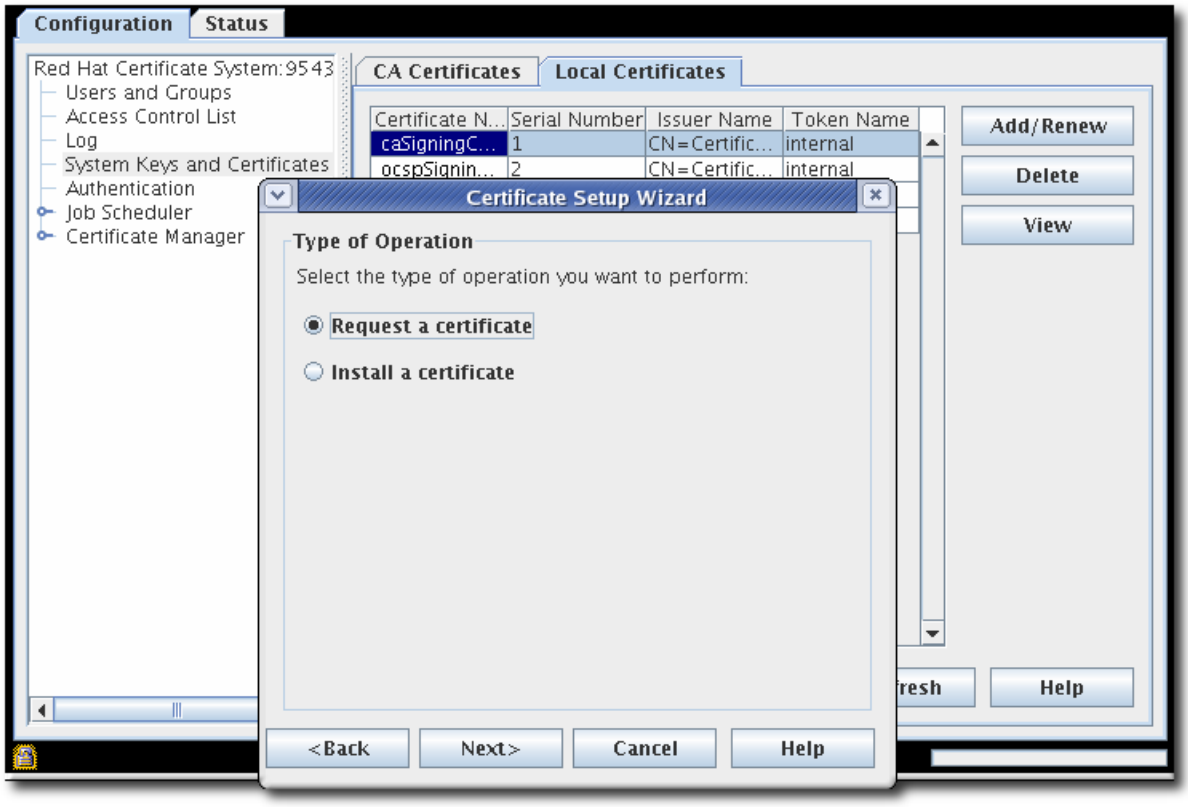
1. 打开子系统控制台。例如：

```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，在导航树中选择 System Keys and Certificates。

3. 在右侧面板中，选择 本地证书 选项卡。

4. 点 Add/Renew。



5.

选择请求证书 单选按钮。

6.

选择请求签名的证书类型。

7.

选择哪个 CA 类型将向请求签名，可以是 root CA 或从属 CA。

8.

设置密钥对信息并设置位置来生成密钥（令牌），可以是内部安全数据库目录或列出的外部令牌之一。

要创建新证书，您必须创建新密钥对。使用现有密钥对将只续订现有证书。

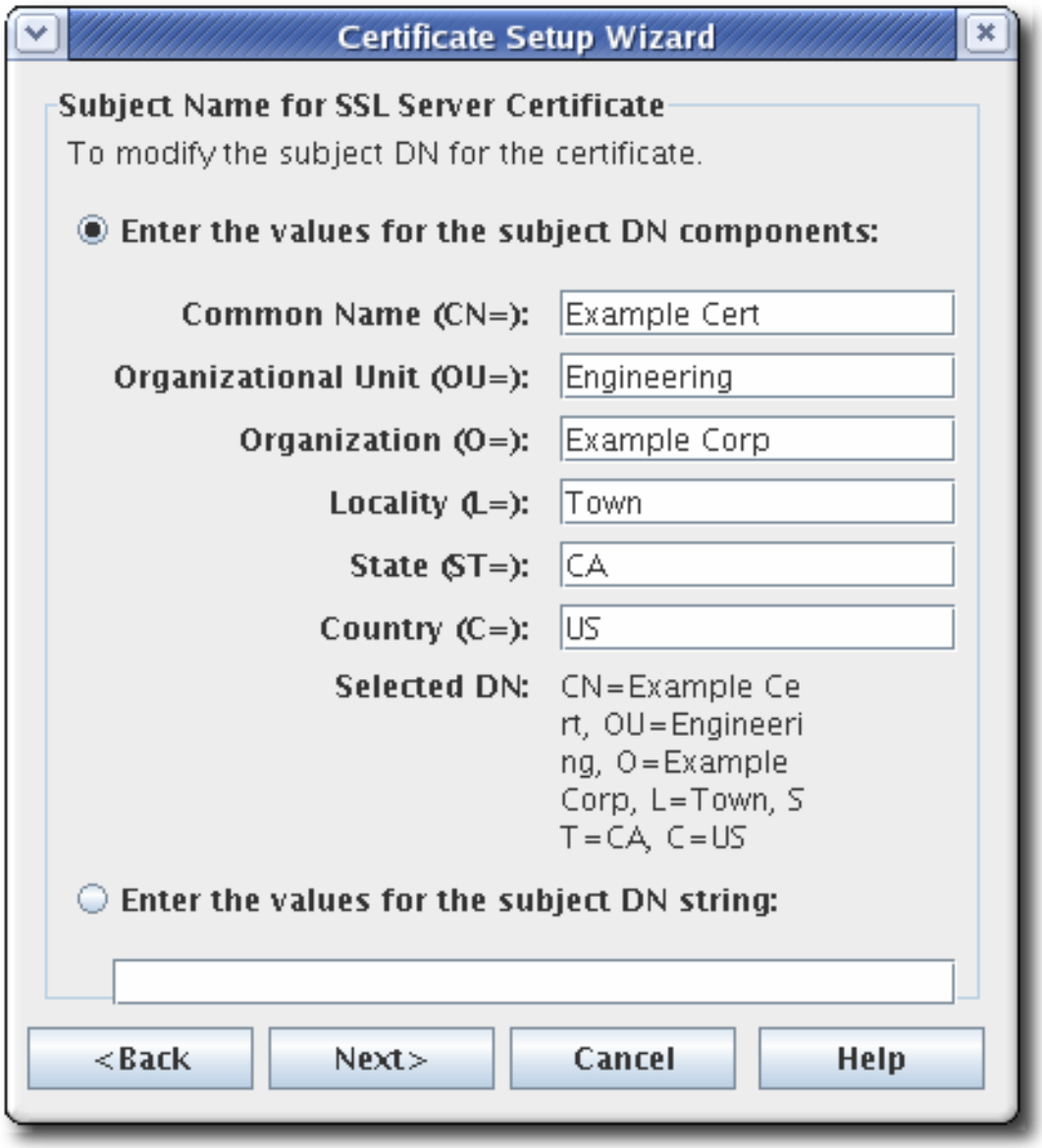

Certificate Setup Wizard
Key-Pair Information for the SSL Server Certificate
Select the token (cryptographic device) on which to generate the key pair:
Token:
internal
Specify which key pair to use for this request:
Use existing key pair
Create new key pair
Specify the key type and key length:
Key type:
RSA
Key length:
1024
bits
1024
2048
Custom
Enter a value
omized key length:
bits
<Back
Next>
Cancel
Help

9.

*选择**消息摘要算法**。*

10.

给主题名称命名。为单独的 DN 属性输入值，以构建主题 DN 或输入完整字符串。

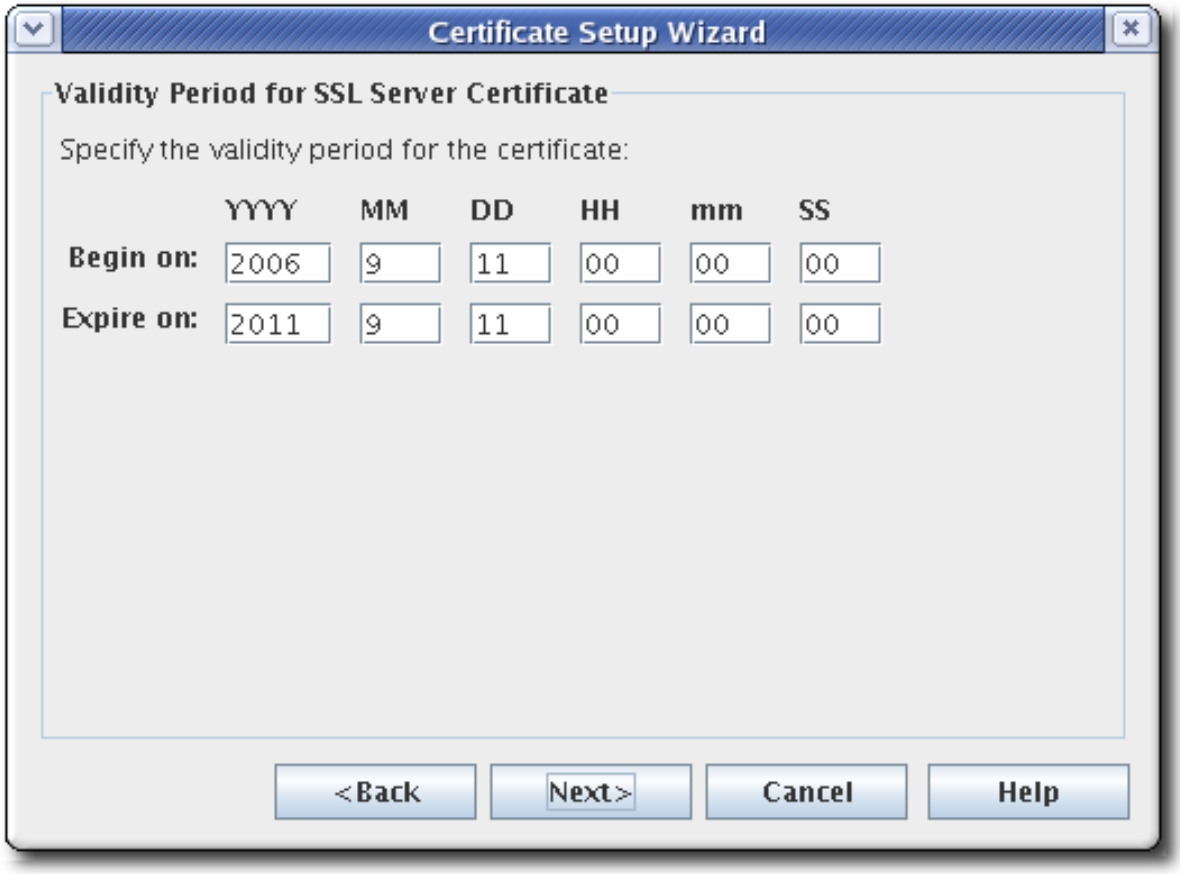


证书请求表单支持通用名称、组织单元和请求者名称字段的所有 UTF-8 字符。

这个支持不包括支持国际化的域名。

11.

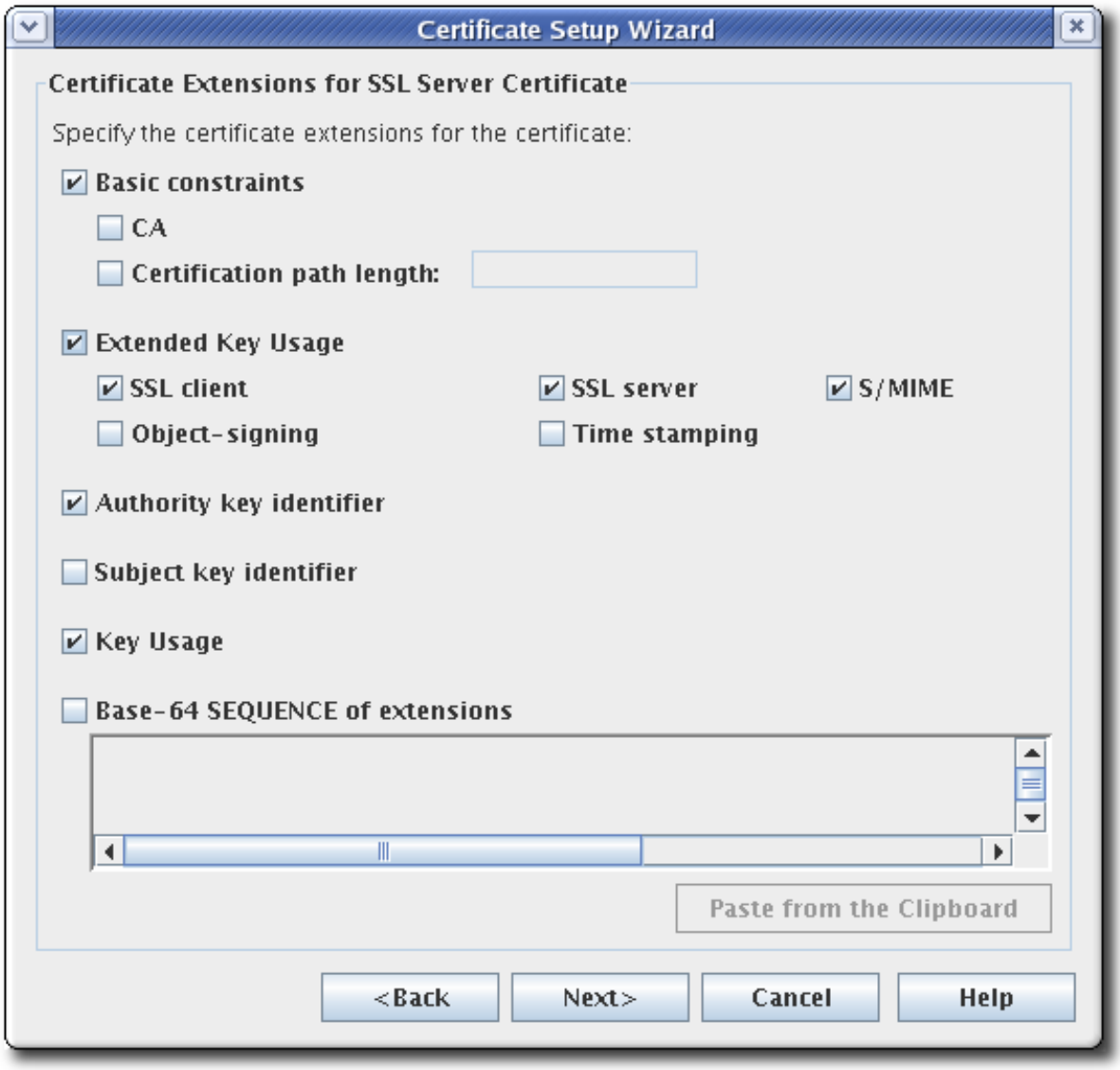
指定证书的有效性期限的开始和结束日期，以及有效期限将在这些日期开始和结束的时间。

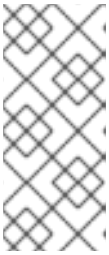


默认有效期为 5 年。

12.

设置证书的标准扩展。默认选择所需的扩展。要更改默认选择，请阅读 附录 B, 证书和 CRL 的默认、限制和扩展 中的说明。





注意

设置 CA 层次结构需要证书扩展。从属 CA 必须具有证书，其中包括将扩展标识为从属 SSL CA（允许他们为 SSL 发出证书）或从属电子邮件 CA（允许他们为安全电子邮件发布证书）。禁用证书扩展意味着无法设置 CA 层次结构。

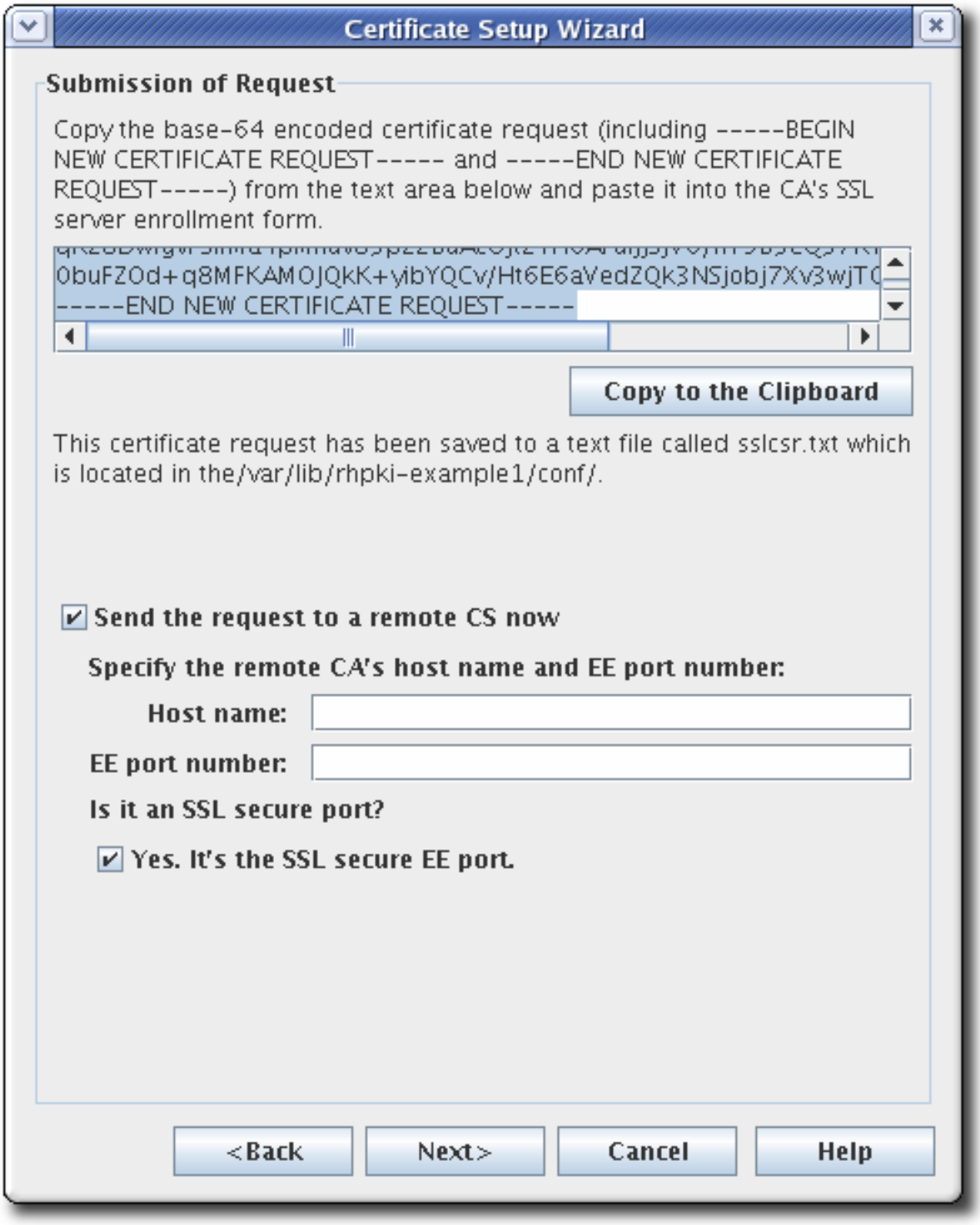
- *基本限制。相关字段是 CA 设置和认证路径长度的数字设置。*
- *扩展密钥使用情况.*
- *授权密钥标识符。*
- *主题键标识符。*

- 关键使用.设定数字签名（位 0）、非推荐（位 1）、密钥证书签名（位 5）和 CRL 符号（位 6）位。该扩展标记为"PKIX 标准"和 RFC 2459 的建议。有关密钥使用扩展的描述，请参阅 RFC 2459。

- 基本64 SEQUENCE.这适用于自定义扩展。将 MIME 64 DER 格式的扩展粘贴到文本字段中。

要添加多个扩展，请使用 ExtJoiner 程序。有关使用工具的详情，请参考 证书系统命令行工具指南。

13. 向导生成密钥对并显示证书签名请求。



请求采用 base-64 编码的 PKCS #10 格式，并通过标记行 -----BEGIN NEW CERTIFICATE REQUEST----- 和 -----END NEW CERTIFICATE REQUEST----- 绑定。例如：

> *-----BEGIN NEW CERTIFICATE REQUEST-----*
> *MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXR*
> *zY2FwZSBDb21tdW5pY2*
> *F0aW9uczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAw*
> *MFoXDTk5MDIyMzE5MDA*
> *wMnbjdgngYoxIDAeBgNVBAoTF05IdHNjYXBIIENvbW11bmljYXRpb25zMQ8wDQYDVQ*
> *QLEwZQZW9wbGUxFz*
> *AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjM*

```
CEGCSqGSIb3Dbndg
JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4
QgEBAQHBAQDAgCAM
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hlBgdR+
jWeLmD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

向导还会将证书请求复制到配置目录中创建的文本文件，该文件位于 /var/lib/pki/instance_name/subsystem_type/conf/ 中。文本文件的名称取决于请求的证书类型。可能的文本文件列在 表 16.1 “为证书签名请求创建的文件” 中。

表 16.1. 为证书签名请求创建的文件

文件名	证书签名请求
cacsr.txt	CA 签名证书
ocspcsr.txt	证书管理器 OCSP 签名证书
ocspcsr.txt	OCSP 签名证书

在将证书请求发送到 CA 之前，不要修改证书请求。该请求可以通过向导自动提交，或复制到剪贴板，并通过其端点页面手动提交到 CA。

注意

向导的自动提交功能只能向远程证书管理器提交请求。它不能用于将请求提交到第三方 CA。要将其提交到第三方 CA，请使用证书请求文件。

14. 检索证书。

    a. 打开证书管理器端点页面。

    ```
    https://server.example.com:8443/ca/ee/ca
    ```

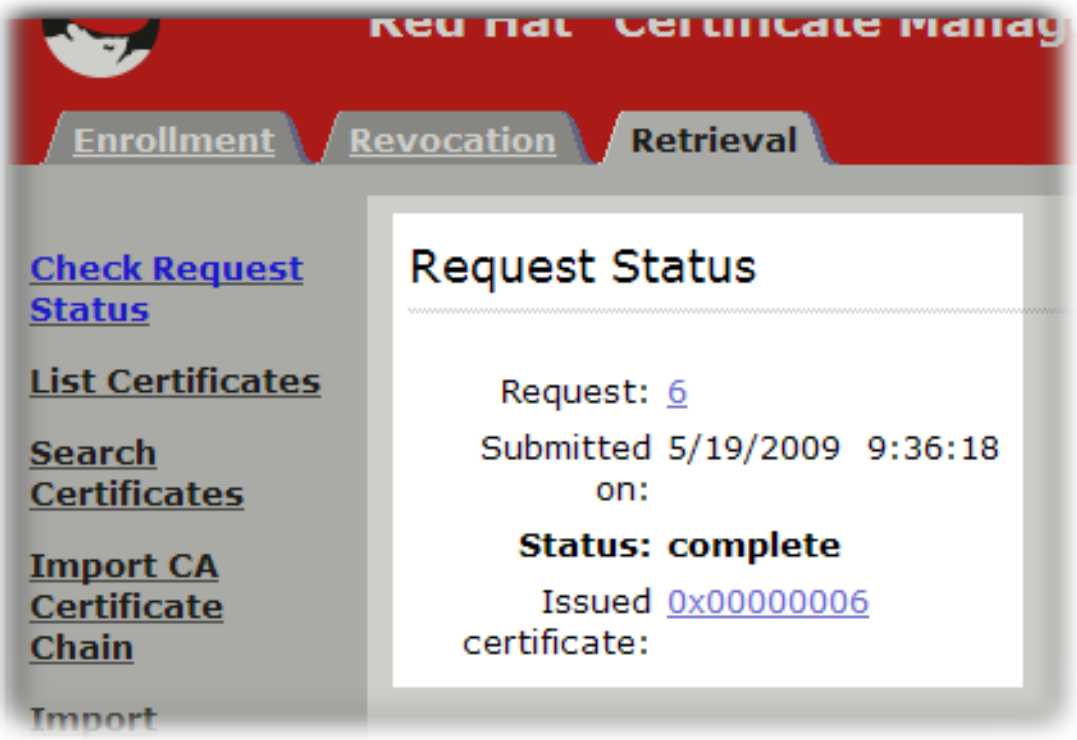
    b. 点 Retrieval 选项卡。

c.

填写提交证书请求时创建的请求 ID 号，然后单击 Submit。

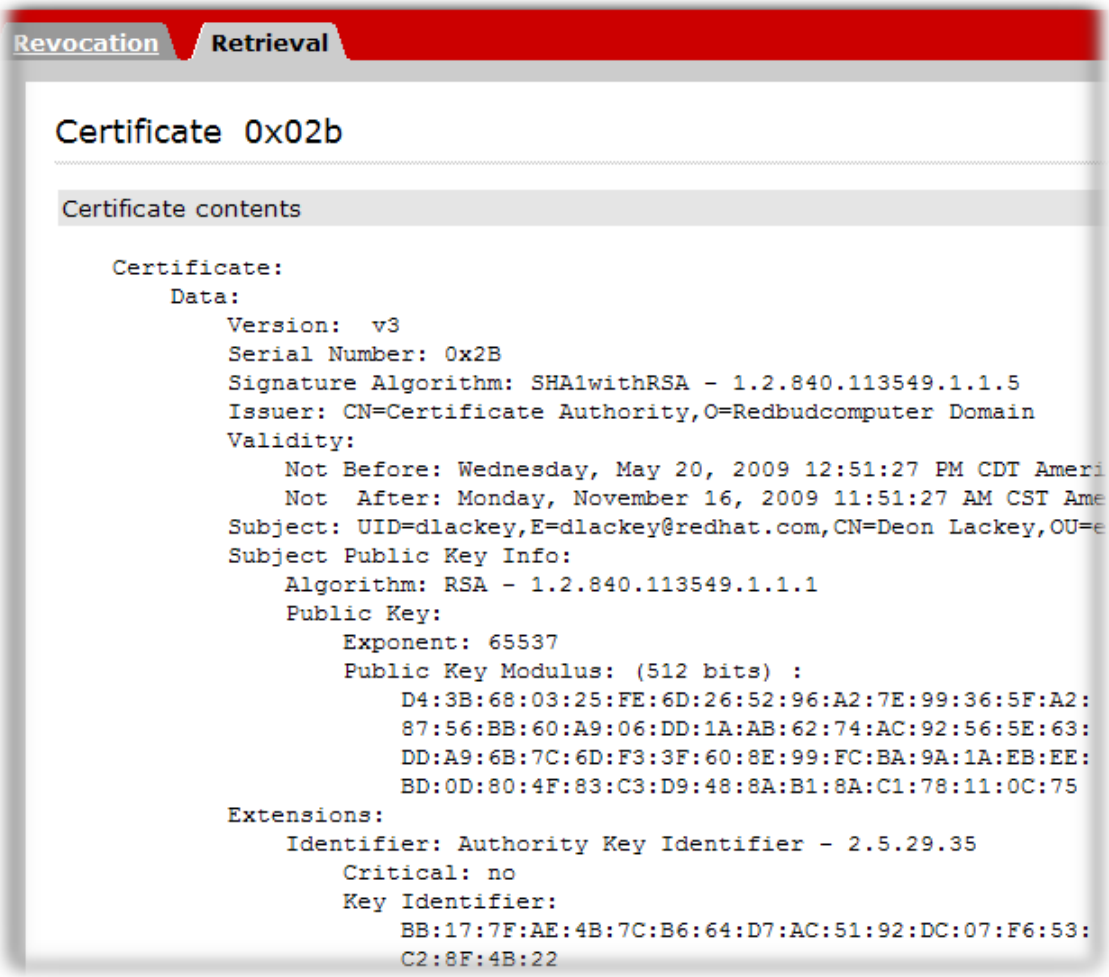
d.

下一页显示证书请求的状态。如果状态已经完成，则证书有一个链接。点签发的证书链接。



e.

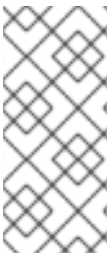
新证书信息以用户用户打印格式显示，采用 base-64 编码格式，采用 PKCS #7 格式。



f.

将 base-64 编码的证书（包括 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE----- 标记行）复制到文本文件。保存文本文件，并使用它来将证书副本存储在子系统的内部数据库中。请参阅 第 14.3.2.1 节 “创建用户”。

16.2.2. 请求其他证书



注意

务必要从计算机生成并提交客户端请求，该请求稍后将用于访问子系统，因为请求过程的一部分在本地计算机上生成私钥。如果需要独立位置，请使用硬件令牌（如智能卡）存储密钥对和证书。

1.

打开子系统控制台。例如：

```
pkiconsole https://server.example.com:8443/ca
```

2.

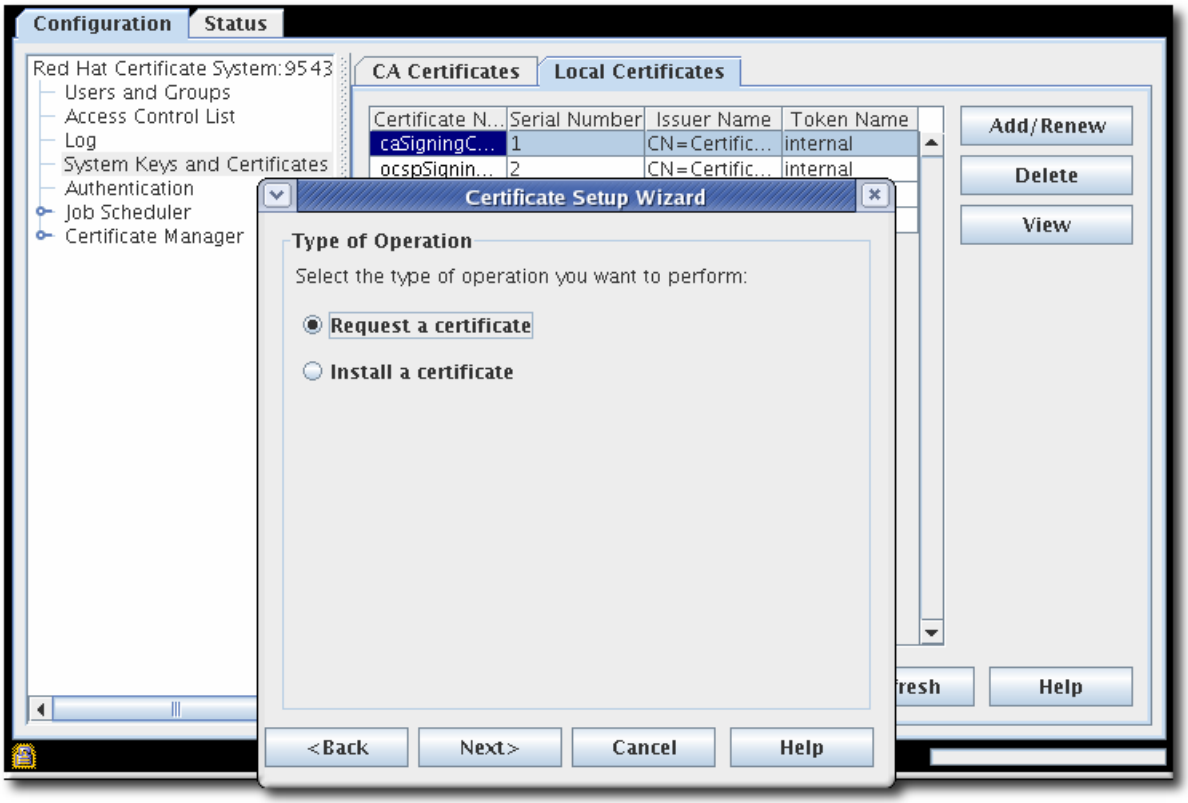
在 Configuration 选项卡中，在导航树中选择 System Keys and Certificates。

3.

在右侧面板中，选择 本地证书 选项卡。

4.

点 Add/Renew。



5.

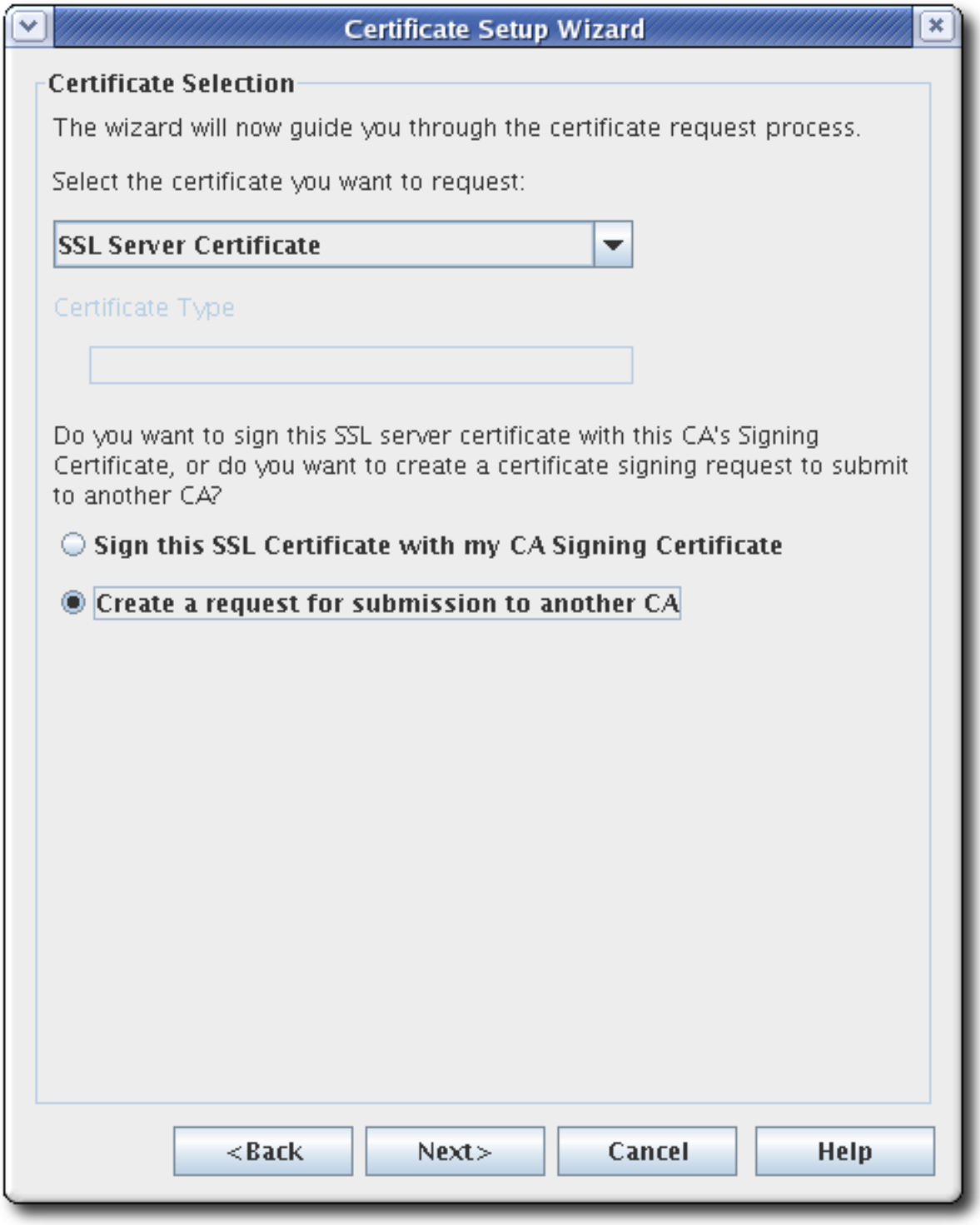
选择请求证书 单选按钮。

6.

选择请求的证书类型。请求的证书的类型因子系统而异。

注意

如果选择创建"其他"证书，则 Certificate Type 字段将变为 active。填写要创建的证书类型，可以是 caCrlSigning，用于 CRL 签名证书的 caSignedLogCert for a 审计日志签名证书，或为 SSL 客户端证书的客户端。



7.

选择哪个 CA 将对请求进行签名。选项是使用本地 CA 签名证书或创建请求提交给另一个 CA。

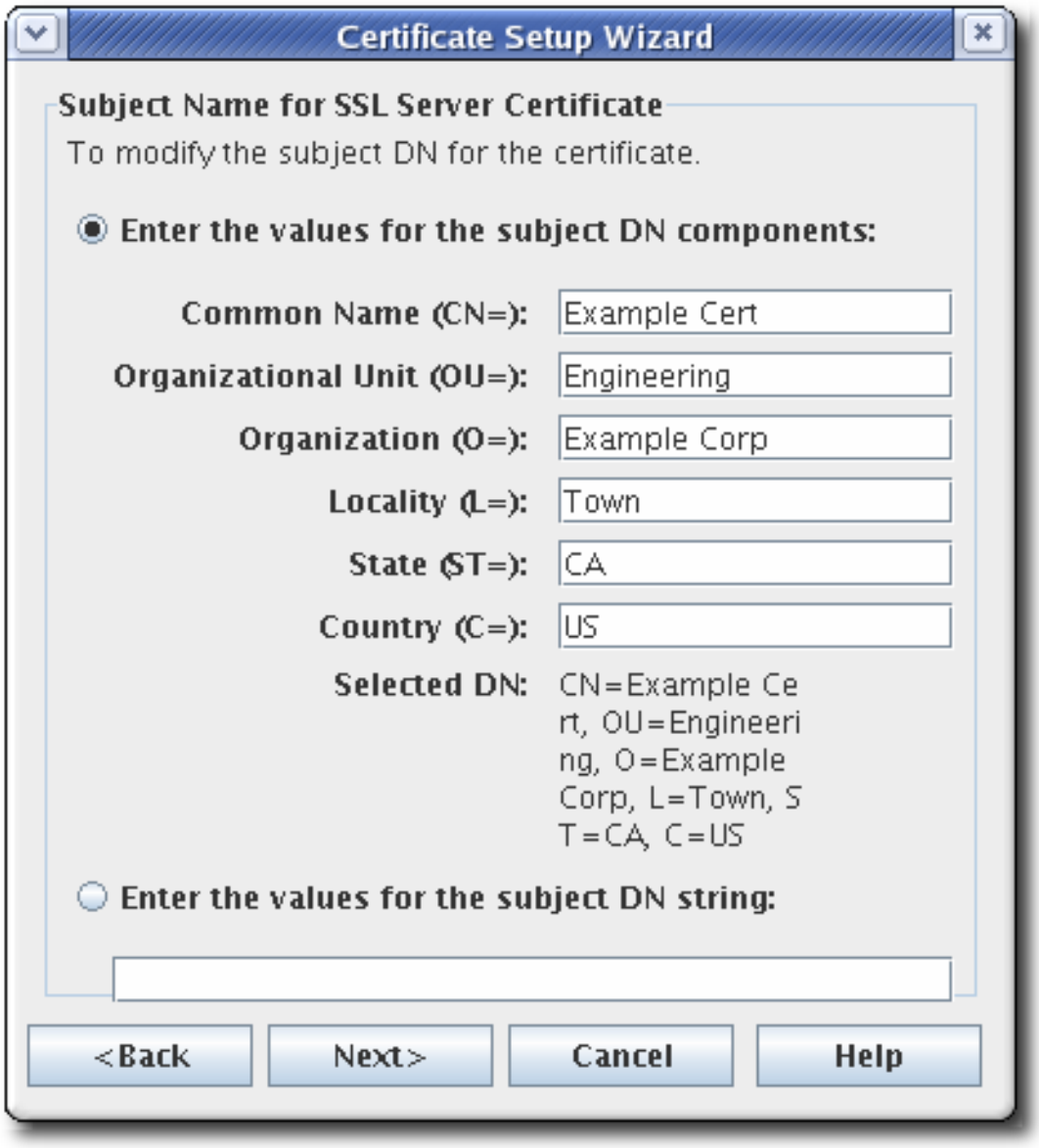
8.

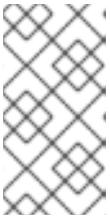
设置密钥对信息并设置位置来生成密钥（令牌），可以是内部安全数据库目录或列出的外部令牌之一。

要创建新证书，您必须创建新密钥对。使用现有密钥对将只续订现有证书。

9.

给主题名称命名。为单独的 DN 属性输入值，以构建主题 DN 或输入完整字符串。





注意

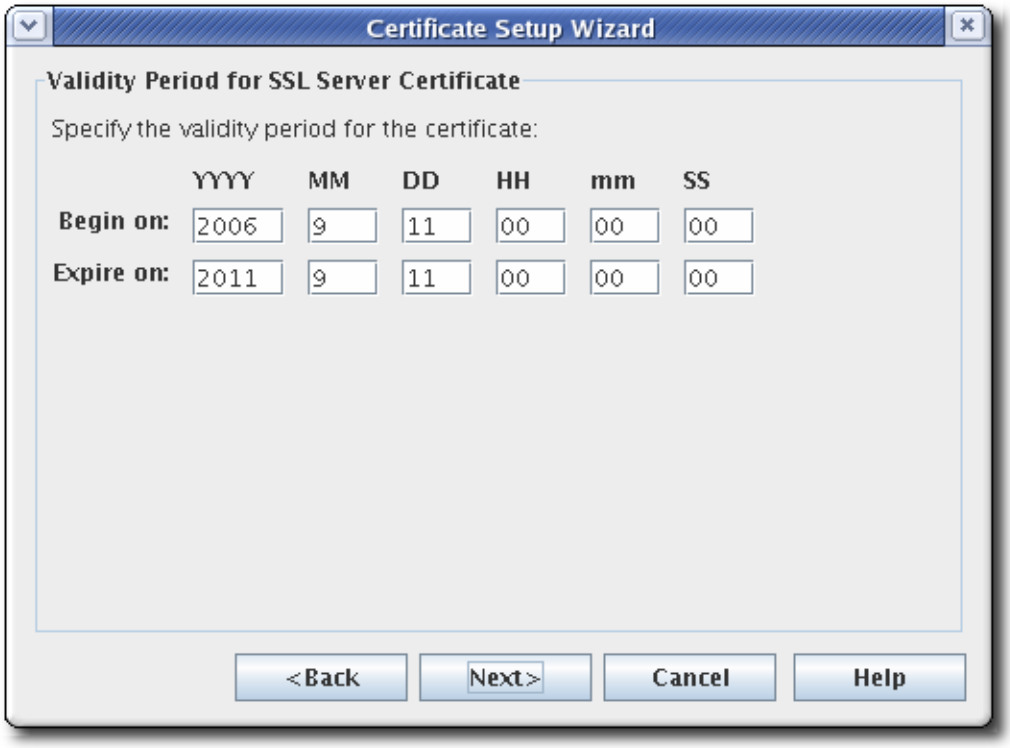
*对于 **SSL** 服务器证书，通用名称必须是证书证书系统**nbsp** 的完全限定主机名；**System**，格式为 **machine_name.domain.domain**。*

***CA** 证书请求表单支持通用名称、组织单元和请求者名称字段的所有 **UTF-8** 字符。*

这个支持不包括支持国际化的域名。

10.

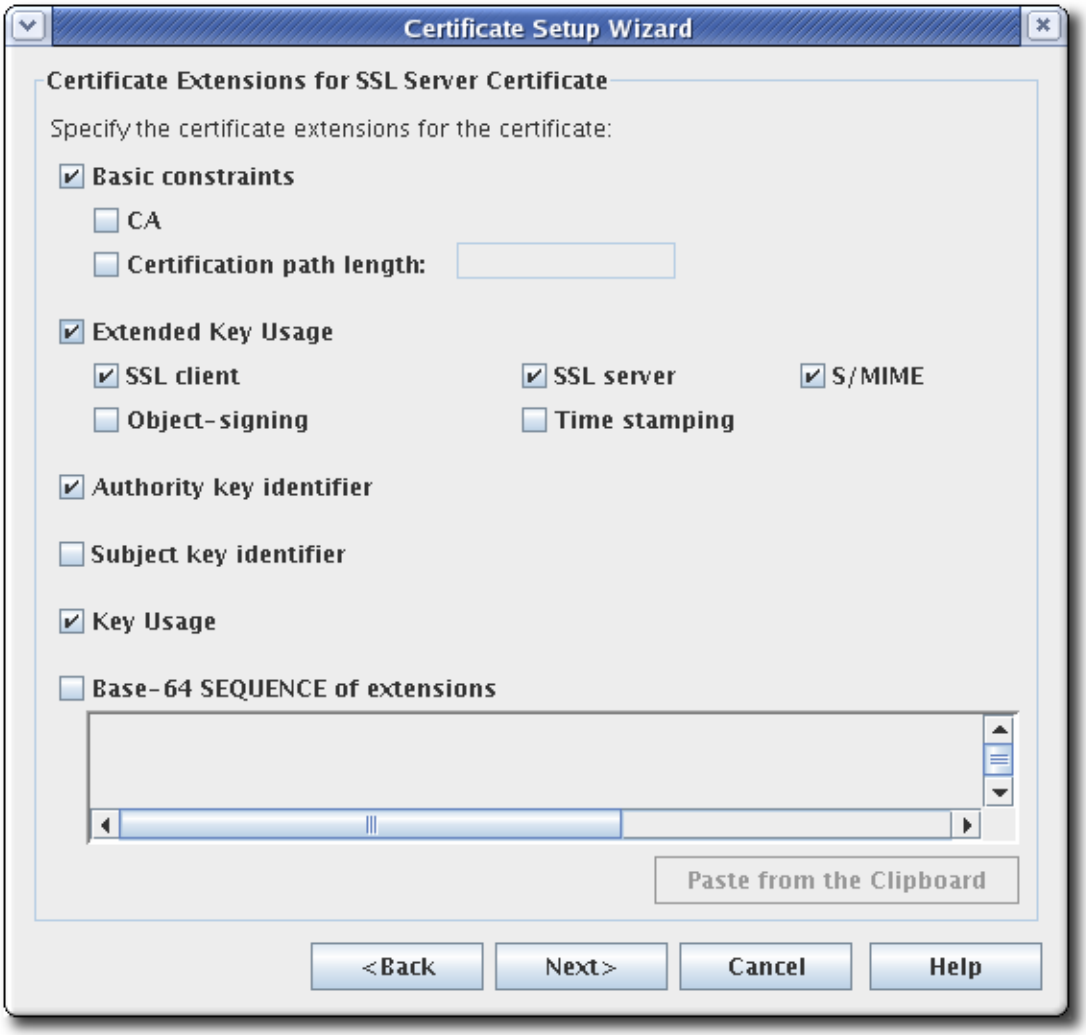
指定证书的有效性期限的开始和结束日期，以及有效期限将在这些日期开始和结束的时间。



默认有效期为 5 年。

11.

设置证书的标准扩展。默认选择所需的扩展。要更改默认选择，请阅读 附录 B, 证书和 CRL 的默认、限制和扩展 中的说明。



- *扩展密钥使用情况.*

- *授权密钥标识符。*

- *主题键标识符。*

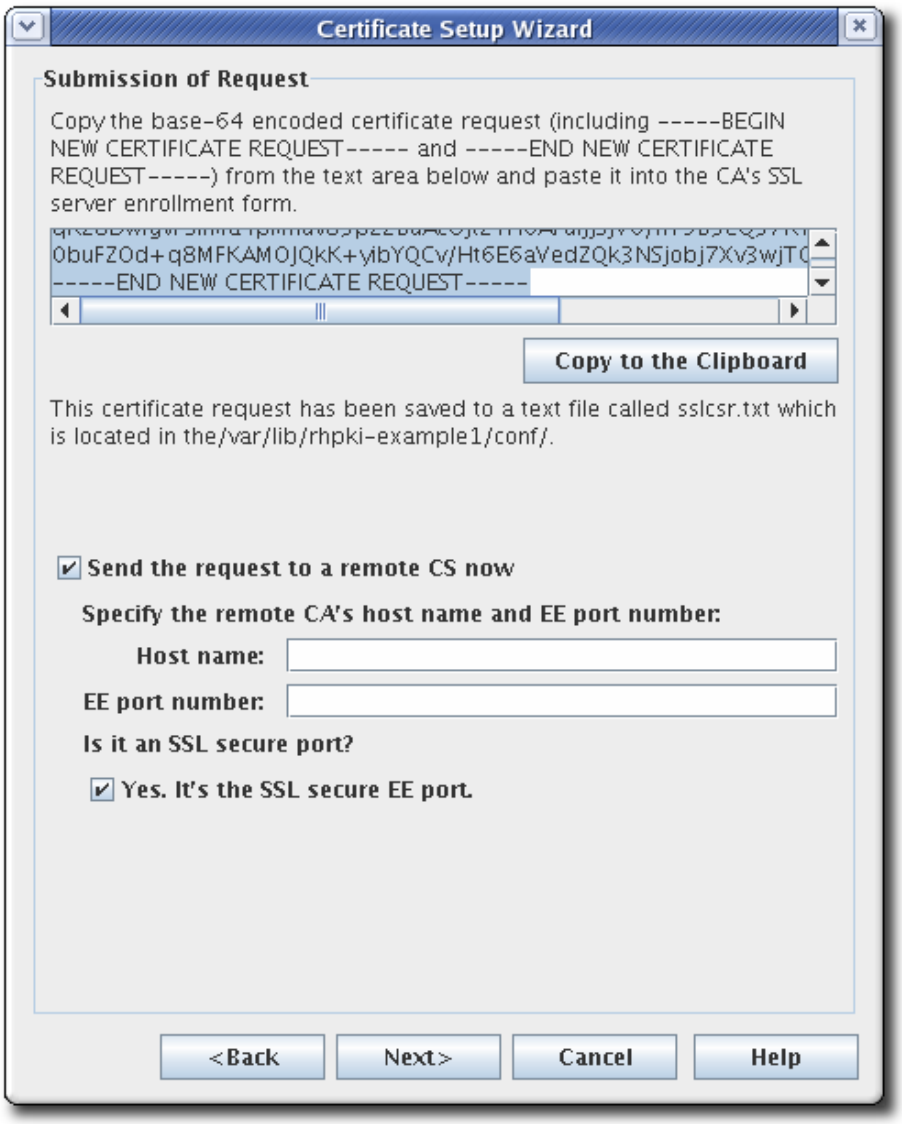
- *关键使用.设定数字签名（位 0）、非推荐（位 1）、密钥证书签名（位 5）和 CRL 符号（位 6）位。该扩展标记为"PKIX 标准"和 RFC 2459 的建议。有关密钥使用扩展的描述，请参阅 RFC 2459。*

- *基本64 SEQUENCE.这适用于自定义扩展。将 MIME 64 DER 格式的扩展粘贴到文本字段中。*

要添加多个扩展，请使用 ExtJoiner 程序。有关使用工具的详情，请参考 证书系统命令行工具指南。

12.

向导生成密钥对并显示证书签名请求。



请求采用 base-64 编码的 PKCS #10 格式，并通过标记行 -----BEGIN NEW CERTIFICATE REQUEST----- 和 -----END NEW CERTIFICATE REQUEST----- 绑定。例如：

> *-----BEGIN NEW CERTIFICATE REQUEST-----*
> *MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXRzY2F*
> *wZSBDb21tdW5pY2*
> *F0aW9ucczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAwMFo*
> *XDTk5MDIyMzE5MDA*
> *wMnbjdgngYoxIDAeBgNVBAoTF05ldHNjYXBlIENvbW11bmljYXRpb25zMQ8wDQYDVQQLEw*
> *ZQZW9wbGUxFz*
> *AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjMCEGC*
> *SqGSIb3Dbndg*
> *JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4QgEB*
> *AQHBAQDAgCAM*

```
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hIBgdR+jWeL
mD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

向导还会将证书请求复制到配置目录中创建的文本文件，该文件位于 /var/lib/pki/instance_name/subsystem_type/conf/ 中。文本文件的名称取决于请求的证书类型。可能的文本文件列在 表 16.2 “为证书签名请求创建的文件” 中。

表 16.2. 为证书签名请求创建的文件

文件名	证书签名请求
kracsr.txt	KRA 传输证书
sslcsr.txt	SSL 服务器证书
othercsr.txt	其他证书，如证书管理器 CRL 签名证书或 SSL 客户端证书

在将证书请求发送到 CA 之前，不要修改证书请求。该请求可以通过向导自动提交，或复制到剪贴板，并通过其端点页面手动提交到 CA。

注意

向导的自动提交功能只能向远程证书管理器提交请求。它不能用于将请求提交到第三方 CA。要将请求提交给第三方 CA，请使用其中一个证书请求文件。

13. 检索证书。

a. 打开证书管理器端点页面。

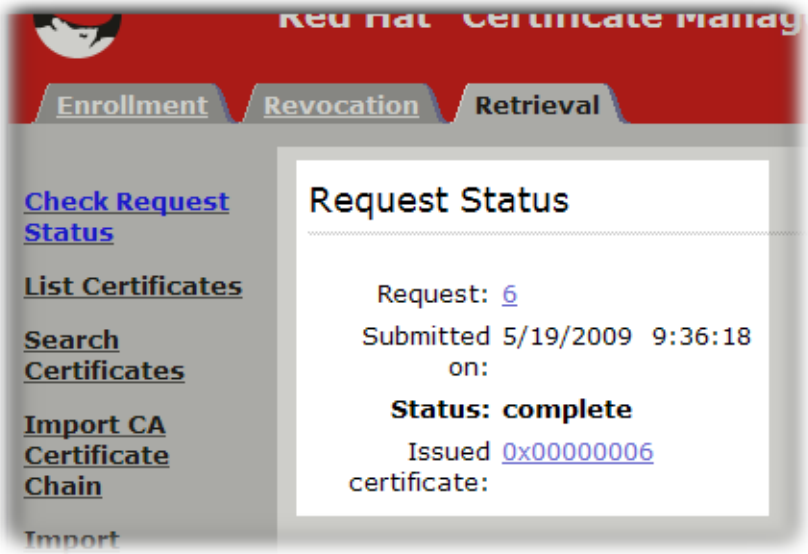
```
https://server.example.com:8443/ca/ee/ca
```

b. 点 Retrieval 选项卡。

c. 填写提交证书请求时创建的请求 ID 号，然后单击 Submit。

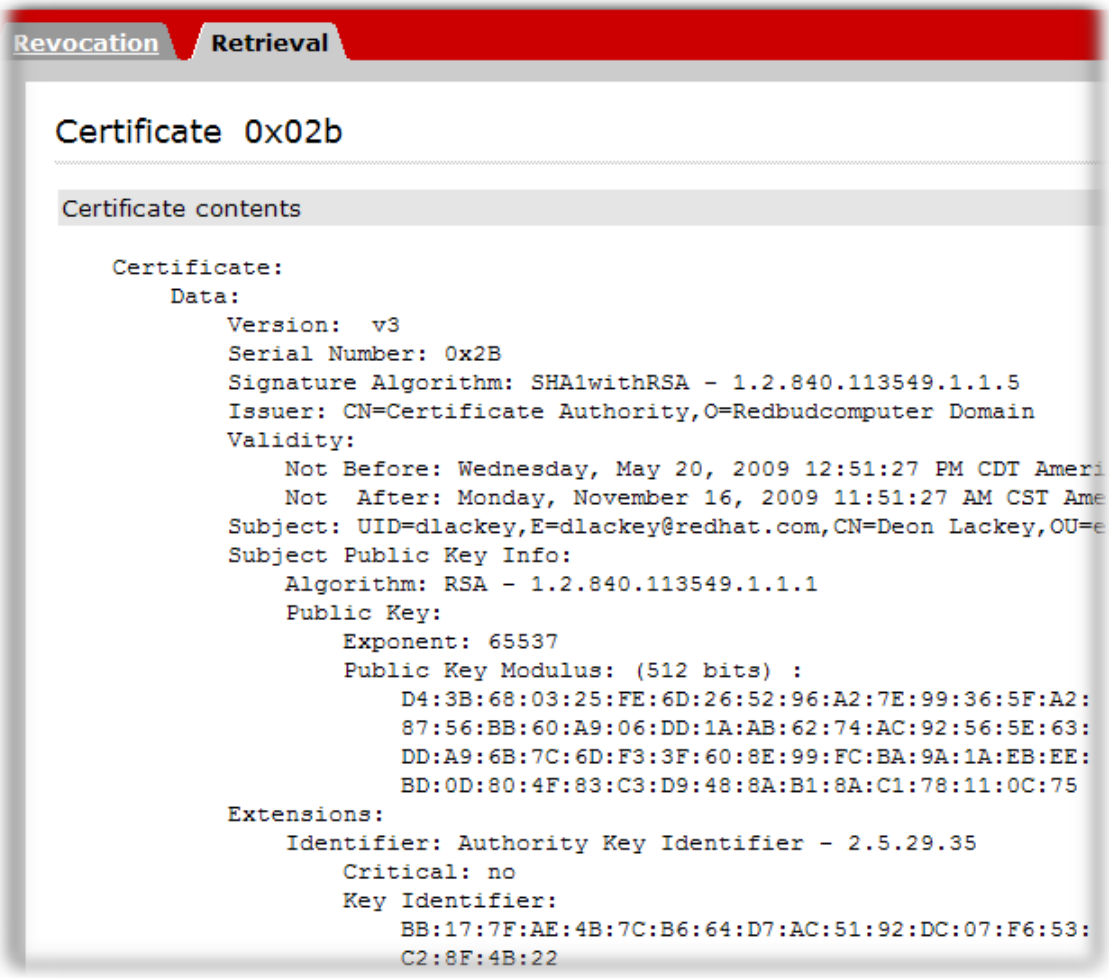
d.

下一页显示证书请求的状态。如果状态已经完成，则证书有一个链接。点签发的证书链接。



e.

新证书信息以用户用户打印格式显示，采用 base-64 编码格式，采用 PKCS #7 格式。



f.

将 base-64 编码的证书（包括 -----BEGIN CERTIFICATE----- 和 -----END

CERTIFICATE----- 标记行）复制到文本文件。保存文本文件，并使用它来将证书副本存储在子系统的内部数据库中。请参阅 第 14.3.2.1 节 “创建用户”。

16.3. 更新 SUBSYSTEM 证书

更新证书的方法有两种。重新生成证书 会提取其原始密钥及其原始配置集并请求，并重新创建相同的密钥，其中包含新的有效期期限和过期日期。重新打包 证书，重新提交到原始配置集的初始证书请求，但生成新的密钥对。管理员可以通过重新密钥来续订管理员证书。

16.3.1. 在 End-Entities Forms 中重新生成证书

通过使用原始证书的序列号，可以在最终用户注册表单中直接续订子系统证书。

1. 按照 第 5.5 节 “续订证书” 中所述，更新 CA 端到端形式的证书。这需要续订子系统证书的序列号。

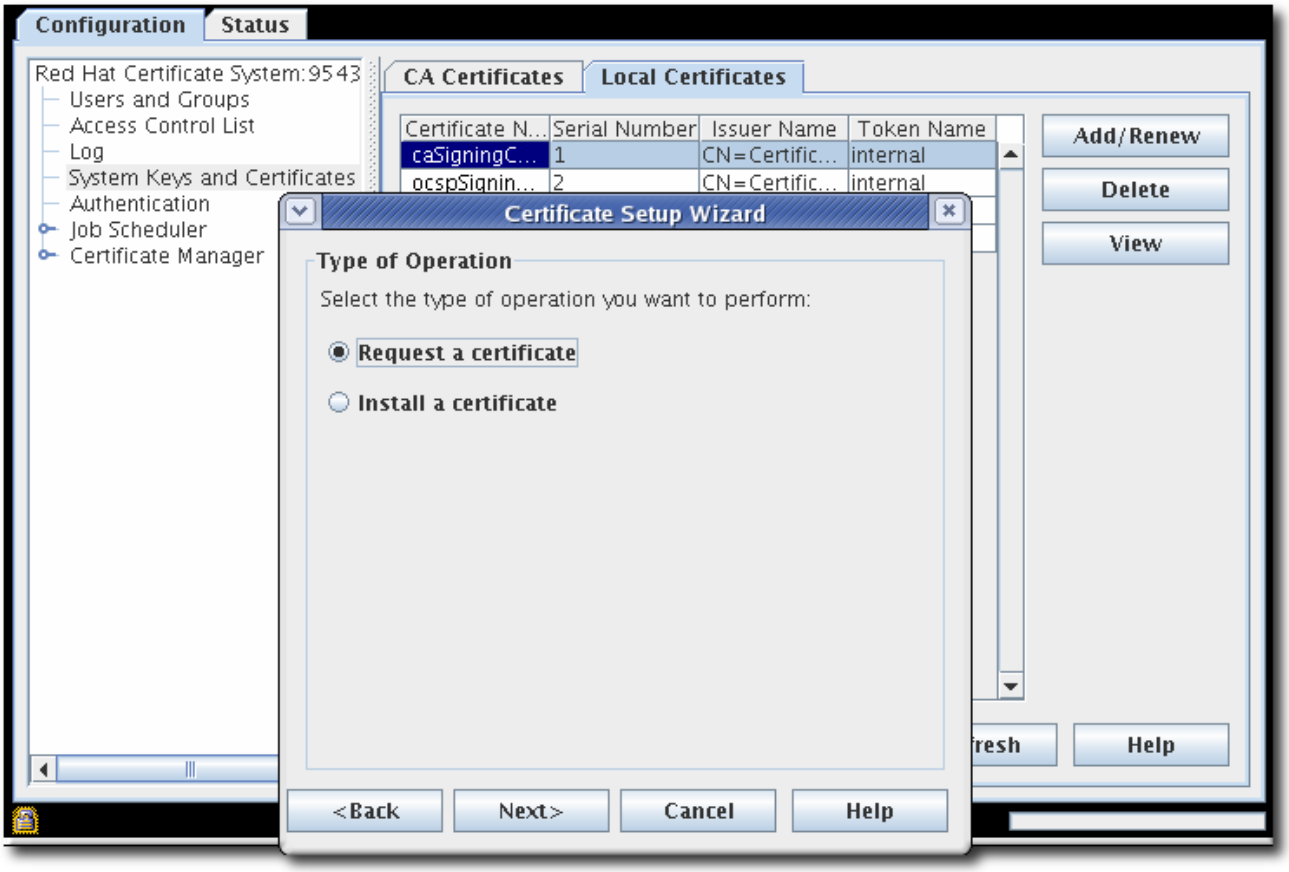
2. 将证书导入到子系统的数据库，如 第 16.6.1 节 “在证书系统数据库中安装证书” 所述。证书可以使用 certutil 或 console 导入。例如：

```
certutil -A -n "ServerCert cert-example" -t u,u,u -d /var/lib/pki/instance_name/alias -a -i
/tmp/example.cert
```

16.3.2. 在控制台中更新证书

Java 子系统可以通过其管理控制台续订任何子系统证书。此过程与请求新子系统证书(第 16.2 节 “通过控制台请求证书”)完全相同，有一个关键区别：续订会使用 现有的密钥对，而不是生成新密钥。

图 16.1. 续订子系统证书



续订证书后，然后从数据库中删除原始证书(第 16.6.3 节 “从数据库中删除证书”)。

16.3.3. 使用 certutil 更新证书

certutil 可用于使用证书数据库中的现有密钥对生成证书请求。然后，新证书请求可以通过 CA 的常规配置集页面提交以签发更新的证书。

注意

加密和签名证书在一个步骤中创建。但是，续订过程一次仅更新一个证书。

要更新证书对中的这两个证书，必须逐一续订每个证书。

1. **获取令牌数据库的密码。**

```
cat /var/lib/pki/instance_name/conf/password.conf

internal=263163888660
```

2. 打开证书被续订的实例的证书数据库目录。

```
cd /var/lib/pki/instance_name/alias
```

3. 列出要续订的证书的密钥和别名。要续订证书，用于生成的密钥对以及提供给新证书的主题名称必须与旧证书中的相同。

```
# certutil -K -d .

certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key and Certificate Services"
Enter Password or Pin for "NSS Certificate DB":
< 0> rsa      69481646e38a6154dc105960aa24ccf61309d37d   caSigningCert cert-pki-tomcat CA
```

4. 复制 别名 目录作为备份，然后从证书数据库中删除原始证书。例如：

```
certutil -D -n "ServerCert cert-example"  -d .
```

5. 运行 **certutil** 命令，并将选项设置为现有证书中的值。

```
certutil -d . -R -n "NSS Certificate DB:cert-pki-tomcat CA" -s "cn=CA Authority,o=Example Domain" -a -o example.req2.txt
```

生成新证书和密钥对与续订证书之间的区别是 **-n** 选项的值。要生成全新的请求和密钥对，而 **-k** 会设置密钥类型，而 **-g** 则用于设置位长度。对于续订请求，**-n** 选项使用证书 **nickname** 来访问存储在安全数据库中的现有密钥对。

有关参数的详情，请查看 **certutil(1) man page**。

6. 提交证书请求，然后检索它并安装它，如 第 5.4 节 “请求和接收证书” 所述。

16.3.4. 更新系统证书

在 PKI 服务器运行时，证书系统不会自动在线更新系统证书。但是，如果系统证书过期，证书系统会无法启动。

更新系统证书：

1. 如果系统证书已过期：

   a. 创建临时证书：

   ```
   # pki-server cert-create sslserver --temp
   ```

   b. 将临时证书导入证书系统的网络安全服务(NSS)数据库中：

   ```
   # pki-server cert-import sslserver
   ```

   c. 启动证书系统：

   ```
   # systemctl start pki-tomcatd@instance_name.service
   ```

2. 显示证书并记录过期系统证书的 ID：

   ```
   # pki-server cert-find
   ```

3. 创建新的永久证书：

   ```
   # pki-server cert-create certificate_ID
   ```

4. 停止证书系统：

   ```
   # systemctl stop pki-tomcatd@instance_name.service
   ```

5. 导入新证书来替换过期的证书：

   ```
   # pki-server cert-import certificate_ID
   ```

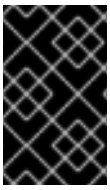
6. 启动证书系统：

```
# systemctl start pki-tomcatd@instance_name.service
```

16.4. 更改子系统证书的名称

更新证书的一种替代方法是用新证书替换它们，这意味着会使用新密钥生成新证书。通常，一个新证书可以添加到数据库和旧的证书，它是一个简单的一对一交换。这是因为单个子系统服务器根据其 nickname 识别证书；只要证书别名保持不变，服务器就可以找到所需的证书，即使主题名称、序列号或密钥一样，该服务器也可以找到所需的证书（如主题名称、序列号或密钥）。

然而，在某些情况下，新证书也可能具有新证书 nickname。在这种情况下，需要在子系统的 CS.cfg 配置文件中的所有必要设置中更新证书 nickname。



重要

在编辑 CS.cfg 文件后，始终重启子系统。

这些表列出了每个子系统证书的所有配置参数：

- 表 16.3 “CA 证书 Nickname 参数”
- 表 16.4 “KRA Certificate Nickname 参数”
- 表 16.5 “OCSP 证书 Nickname 参数”
- 表 16.6 “TKS 证书 Nickname 参数”
- 表 16.7 “CS.cfg 中的 TPS Nickname 参数”

表 16.3. CA 证书 Nickname 参数

CA 签署证书	• ca.cert.signing.nickname • ca.signing.cacertnickname • ca.signing.certnickname • ca.signing.nickname • cloning.signing.nickname
OCSP 签署证书	• ca.ocsp_signing.cacertnickname • ca.ocsp_signing.certnickname • ca.cert.ocsp_signing.nickname • ca.ocsp_signing.nickname • cloning.ocsp_signing.nickname
Subsystem Certificate	• ca.cert.subsystem.nickname • ca.subsystem.nickname • cloning.subsystem.nickname • pkiremove.cert.subsystem.nickname
服务器证书	• ca.sslserver.nickname • ca.cert.sslserver.nickname
审计签署证书	• ca.audit_signing.nickname • ca.cert.audit_signing.nickname • cloning.audit_signing.nickname

表 16.4. KRA Certificate Nickname 参数

传输证书	• cloning.transport.nickname • kra.cert.transport.nickname • kra.transport.nickname • tks.kra_transport_cert_nickname 请注意，此参数位于 TKS 配置文件中。如果 *KRA* 传输证书 nickname 发生变化，这需要在 TKS 配置中更改，即使 TKS 证书都保持不变。
Storage Certificate	• cloning.storage.nickname • kra.storage.nickname • kra.cert.storage.nickname
服务器证书	• kra.cert.sslserver.nickname • kra.sslserver.nickname
Subsystem Certificate	• cloning.subsystem.nickname • kra.cert.subsystem.nickname • kra.subsystem.nickname • pkiremove.cert.subsystem.nickname
审计日志签名证书	• cloning.audit_signing.nickname • kra.cert.audit_signing.nickname • kra.audit_signing.nickname

表 16.5. OCSP 证书 Nickname 参数

OCSP 签署证书	• cloning.signing.nickname • ocsp.signing.certnickname • ocsp.signing.cacertnickname • ocsp.signing.nickname

服务器证书	• ocsp.cert.sslserver.nickname • ocsp.sslserver.nickname
Subsystem Certificate	• cloning.subsystem.nickname • ocsp.subsystem.nickname • ocsp.cert.subsystem.nickname • pkiremove.cert.subsystem
审计日志签名证书	• cloning.audit_signing.nickname • ocsp.audit_signing.nickname • ocsp.cert.audit_signing.nickname

表 16.6. TKS 证书 Nickname 参数

KRA 传输证书[a]	• tks.kra_transport_cert_nickname
服务器证书	• tks.cert.sslserver.nickname • tks.sslserver.nickname
Subsystem Certificate	• cloning.subsystem.nickname • tks.cert.subsystem.nickname • tks.subsystem.nickname • pkiremove.cert.subsystem.nickname
审计日志签名证书	• cloning.audit_signing.nickname • tks.audit_signing.nickname • tks.cert.audit_signing.nickname
[a] 如果 *KRA* 传输证书 nickname 发生变化，这需要在 TKS 配置中更改，即使 TKS 证书都保持不变。	

表 16.7. CS.cfg 中的 TPS Nickname 参数

服务器证书	• tps.cert.sslserver.nickname
Subsystem Certificate	• tps.cert.subsystem.nickname • selftests.plugin.TPSValidity.nickname • selftests.plugin.TPSPresence.nickname • pkiremove.cert.subsystem.nickname
审计日志签名证书	• tps.cert.audit_signing.nickname

16.5. 使用跨PAIR 证书

在上世纪九十年代，由于美国政府开始增强其公钥基础架构，它很明显，政府的分支也很明显，仍需要单独的 PKI 部署来识别和信任其他证书，就像 证书由自己的 CA 发布一样。（让外部客户端信任的证书在网络外信任的证书是非常严重的，无法轻松解决任何 PKI 管理员的问题。）

美国政府制定了标准，用于发布 跨对证书，称为联邦网桥证书颁发机构。由于明显的原因，这些证书也称为网桥证书。网桥或跨对证书是作为双证书对帧的 CA 签名证书，类似于用户的加密和签名证书对，只有对中的每个证书由不同的 CA 发布。两个合作伙伴 CA 都在其数据库中存储其他 CA 签名证书，因此其他 PKI 中发布的所有证书都是可信且被识别的。

桥接证书遵循由 CA 发布的证书，这些证书没有链接到自己的 PKI 中的 root CA。通过在证书证书证书间建立信任;System CA 和另一个 CA 通过跨对 CA 证书进行下载，从而下载并用于信任由其他 CA 发布的证书，只需下载并安装 CA 证书信任所有证书。

CertificateCertificate Systemnbsp;System 可以发出、导入和发布跨对 CA 证书。必须创建一个特殊的配置集，以发布跨对证书，然后使用 CA 子系统的证书向导来请求并安装 CA。

有关创建跨对证书配置集的更多信息，请参阅 红帽认证系统 9 规划、安装和部署指南中的 配置跨Pair 配置集 部分。

有关发布跨对证书的更多信息，请参阅 第 8.9 节 “发布跨Pair 证书”。

16.5.1. 安装跨Pair 证书

跨对证书都可以导入到证书Certificate Systemnbsp 中；使用 certutil 工具进行系统数据库，也可以通过从证书设置向导中选择 Cross-Pair Certificates 选项，如 第 16.6.1 节 “在证书系统数据库中安装证书” 所述。

当两个证书都导入到数据库中后，crossCertificatePair 条目会被创建并存储在数据库中。创建 crossCertificatePair 条目后，原始的跨对 CA 证书被删除。

16.5.2. 搜索跨Pair 证书

网桥证书中的两个 CA 都可以在 LDAP 数据库中存储或发布 跨证书作为跨CertificatePair 条目。可以使用 ldapsearch 搜索证书管理器的内部数据库，以查找 crossCertificatePair 条目。

```
/usr/lib[64]/mozldap/ldapsearch -D "cn=directory manager" -w secret -p 389 -h
server.example.com -b "o=server.example.com-pki-ca" -s sub "(crossCertificatePair=*)"
```

16.6. 管理证书数据库

每个证书证书系统启动;System 实例有一个证书数据库，该数据库在其内部令牌中维护。此数据库包含属于在 CertificateCertificate Systemnbsp 中安装的子系统的证书；System instance 和各种 CA 证书，用于验证它们接收的证书。

即使外部令牌用于生成和存储密钥对，CertificateCertificate Systemnbsp;System 始终在其内部令牌中维护其可信和不可信 CA 证书的列表。

本节介绍如何使用 CertificateCertificate Systemnbsp;System 窗口查看证书数据库的内容、删除不需要的证书以及更改数据库中安装的 CA 证书的信任设置。有关在数据库中添加证书的详情请参考 第 16.6.1 节 “在证书系统数据库中安装证书”。

注意

CertificateCertificate Systemnbsp;System 命令行实用工具 certutil 可以用来通过编辑信任设置和删除证书来管理证书数据库。有关这个工具的详情，请参考 http://www.mozilla.org/projects/security/pki/nss/tools/。

管理员应定期检查证书数据库的内容，以确保它不会包含任何不需要的 CA 证书。例如，如果数据库包含不应在 PKI 设置中不应信任的 CA 证书，请删除它们。

16.6.1. 在证书系统数据库中安装证书

如果为子系统发布新的服务器证书，则必须将其安装到该子系统数据库中。此外，必须在子系统数据库中安装用户和代理证书。如果证书由外部 CA 发布，则通常需要安装对应的 CA 证书或证书链。

证书可以通过控制台的证书设置向导或使用 certutil 工具安装在子系统证书数据库中。

- 第 16.6.1.1 节 “通过控制台安装证书”
- 第 16.6.1.2 节 “使用 certutil 安装证书”
- 第 16.6.1.3 节 “关于 CA 证书链”

16.6.1.1. 通过控制台安装证书

证书设置向导可以安装或将以下证书导入到内部或者由 CertificateCertificate Systemnbsp;System 实例使用的外部令牌中：

- CertificateCertificate Systemnbsp;System 子系统使用的任何证书
- 来自外部 CA 或其他证书证书系统nbsp 的任何可信 CA 证书;系统 CA
- 证书链

证书链包括一组证书：主题证书、可信根 CA 证书以及将主题证书链接到可信根所需的任何中间 CA 证书。但是，向导导入的证书链必须只包含 CA 证书；任何证书都不能是一个用户证书。

在证书链中，链中的每个证书都被编码为一个独立的 DER 编码对象。当向导导入证书链时，它会在另一个对象后导入这些对象，它们的所有方法是链到最后一个证书，这可能不是 root CA 证书。如果链中的任何证书已在本地证书数据库中安装，向导会将现有证书替换为链中的现有证书。如果链包含中间 CA 证书，向导会将它们添加到证书数据库中，作为 不信任的 CA 证书。

子系统控制台使用相同的向导来安装证书和证书链。要在本地安全数据库中安装证书，请执行以下操作：

1. 打开控制台。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

2. 在 Configuration 选项卡中，从左侧导航树中选择 System Keys and Certificates。

3. 有两个标签可以安装证书，具体取决于子系统类型和证书类型。

- CA 证书 标签页是安装 CA 证书和证书链。对于证书管理器，此选项卡用于第三方 CA 证书或其他证书证书系统，系统 CA 证书；所有本地 CA 证书均安装在本地 证书 选项卡中。对于所有其他子系统，所有 CA 证书和链都通过此标签页安装。

- 本地证书 标签页是在安装所有服务器证书、子系统证书和本地证书（如 OCSP 签名或 KRA 传输）的位置。

选择相应的选项卡。

4. 要在 本地证书选项卡中安装证书，请单击 Add/Renew。要在 CA 证书选项卡中安装证书，点 添加。两者都会打开证书设置向导。

a. 当向导打开后，选择" 安装证书 "单选按钮，然后单击" 下一步"。

b. 选择要安装的证书类型。下拉菜单选项与用于创建证书的选项相同，具体取决于子系统类型，还有更多选项来安装跨对证书。

c. 粘贴到证书正文中，包括 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE-----，并将其指定到文本区域，或者指定绝对文件位置；这必须是本地文件。

证书类似如下：

```
-----BEGIN CERTIFICATE-----
```

```
MIICKzCCAZSgAwIBAgIBAzANgkqkiG9w0BAQQFADA3MQswCQYDVQQGEw
JVUzERMA8GA1UEChMITmV0c2NhcGUxFTATBgNVBAsTDFN1cHJpeWEncy
BDQTAeFw05NzEwMTgwMTM2MjVaFw05OTEwMTgwMTM2MjVaMEgxCzAJBg
NVBAYTAlVTMREwDwYDVQQKEwhOZXRzY2FwZTENMAsGA1UECxMEUHawcz
EXMBUGA1UEAxMOU3VwcmI5YSBTaGV0dHkwgZ8wDQYJKoZIhdfNAQEBBQ
ADgY0AMIGJAoGBAMr6eZiPGfjX3uRJgEjmKiqG7SdATYzBcABu1AVyd7
chRFOGD3wNktbf6hRo6EAmM5R1Askzf8AW7LiQZBcrXpc0k4du+2j6xJ
u2MPm8WKuMOTuvzpo+SGXeImHVChEqooCwfdiZywyZNmgaMa2MS6pUkf
QVAgMBAAGjNjA0MBEGCWCGSAGG+EIBAQQEAwIAgD
-----END CERTIFICATE-----
```

5. 向导显示证书详情。查看指纹，以确保这是正确的证书，或使用 Back 按钮返回并提交不同的证书。为证书指定 nickname。

   向导会安装证书。

6. 签署证书的任何 CA 都必须被子系统信任。确保该 CA 的证书存在于子系统的证书数据库中（内部或外部）以及它被信任。

   如果没有列出 CA 证书，请将证书作为可信 CA 添加到证书数据库中。如果列出了 CA 的证书但不受信任的，请将信任设置为可信，如 第 16.7 节 “更改 CA 证书的信任设置” 所示。

   安装未存储在证书证书系统的 CA 发布的证书时，将 CA 的证书链添加到数据库。要将 CA 链添加到数据库，请将 CA 链复制到文本文件，再次启动向导，并安装 CA 链。

16.6.1.2. 使用 certutil 安装证书

要在 CertificateCertificate Systemnbsp 中安装子系统证书；系统实例的安全数据库使用 certutil，请执行以下操作：

1. 打开子系统的安全数据库目录。

   ```
   cd /var/lib/pki/instance_name/alias
   ```

2. 使用 -A 运行 certutil 命令以添加证书，并指向包含 CA 发布的证书的文件。

   ```
   certutil -A -n cert-name -t trustargs
     -d . -a -i certificate_file
   ```

注意

如果 CertificateCertificate Systemnbsp;System instance 的证书和密钥存储在 HSM 中，则使用 -h 选项指定令牌名称。

例如：

```
certutil -A -n "ServerCert cert-instance_name" -t u,u,u -d . -a -i /tmp/example.cert
```

有关使用 certutil 命令的详情请参考 http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

16.6.1.3. 关于 CA 证书链

任何支持证书的客户端或服务器软件都会在其证书数据库中维护一系列可信 CA 证书。这些 CA 证书决定了软件可以验证哪些其他证书。在最简单的情形中，软件只能验证其有证书的其中一个 CA 发布的证书。可信 CA 证书也可能是 CA 证书链的一部分，每个 CA 在证书层次结构中签发。

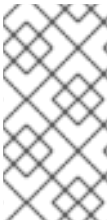
链中的第一个证书以特定于上下文的方式进行处理，具体根据它的导入方式而有所不同。对于 Mozilla Firefox，此处理取决于在正在下载的对象上使用的 MIME 内容类型。对于 Red Hatnbsp;Hat 服务器，它取决于服务器管理界面中选择的选项。

后续证书都相同。如果证书包含 Netscape 证书类型证书扩展中的 SSL-CA 位，且本地证书数据库中尚不存在，则它们会添加为不信任的 CA。只要链中有一个可信的 CA，它们可以用于证书链验证。

16.6.2. 查看数据库内容

可以通过子系统管理控制台查看存储在子系统证书数据库 cert8.db 中的证书。另外，也可以使用 certutil 工具列出证书。必须使用 certutil 来查看 TPS 证书，因为 TPS 子系统不使用管理控制台。

- 第 16.6.2.1 节 “通过控制台查看数据库内容”
- 第 16.6.2.2 节 “使用 certutil 查看数据库内容”



注意

cert8.db 数据库中列出的证书是用于子系统操作的子系统证书。用户证书会存储在 LDAP 内部数据库中的用户条目。

16.6.2.1. 通过控制台查看数据库内容

要通过管理控制台查看数据库的内容，请执行以下操作：

1. **打开子系统控制台。**

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

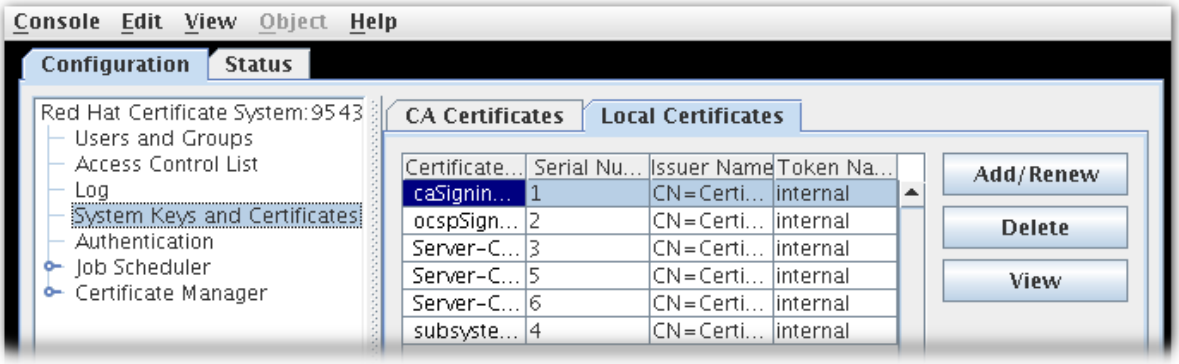
2. **在 Configuration 选项卡中，从左侧导航树中选择 System Keys and Certificates。**

3. **有两个标签页，即 CA 证书和密钥，它们列出不同类型的证书。**

   - **CA 证书列出了对应私钥材料不可用的 CA 证书，如第三方 CA 发布的证书，如 Entrust 或 Verisign 或外部证书证书证书；系统证书管理器；系统证书管理器。**

   - **本地证书 列出了证书证书，由 CertificateCertificate Systemnbsp;System 子系统实例（如 KRA 传输证书或 OCSP 签名证书）保留的证书。**

图 16.2. 证书数据库标签



4. **Certificate Database Management 表列出了子系统上安装的所有证书。每个证书都提供了以下信息：**

- 证书名称
- 序列号
- 签发者名称，是此证书颁发者的通用名称(cn)。
- 令牌名称，存放证书的加密密钥的名称；对于存储在数据库中的证书，这是 内部的。

要查看有关证书的更多详细信息，请选择证书，然后点 View。这会打开一个窗口，显示证书的序列号、有效期周期、主题名称、签发者名称和证书指纹。

16.6.2.2. 使用 certutil 查看数据库内容

要使用 certutil 查看子系统数据库中的证书，打开实例的证书数据库目录，并使用 -L 选项运行 certutil。例如：

```
cd /var/lib/pki/instance_name/alias

certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

要使用 certutil 查看存储在子系统数据库中的密钥，请使用 -K 选项运行 certutil。例如：

```
cd /var/lib/pki/instance_name/alias

certutil -K -d .

Enter Password or Pin for "NSS Certificate DB":
<0> subsystemCert cert-instance_name
<1>
<2> Server-Cert cert-instance_name
```

有关使用 certutil 命令的详情请参考 http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

16.6.3. 从数据库中删除证书

删除不需要的证书会减少证书数据库的大小。

注意

当从证书数据库中删除 CA 证书时，请注意不要删除 中间 CA 证书，这将有助于子系统链到可信的 CA 证书。如果不确定，请将数据库中的证书保留为 不可信 的 CA 证书；请参阅 第 16.7 节 “更改 CA 证书的信任设置”。

- 第 16.6.3.1 节 “通过控制台删除证书”
- 第 16.6.3.2 节 “使用 certutil 删除证书”

16.6.3.1. 通过控制台删除证书

要通过控制台删除证书，请执行以下操作：

1. 打开子系统控制台。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

2. 在 Configuration 选项卡中，从左侧导航树中选择 System Keys and Certificates。
3. 选择要删除的证书，然后点 Delete。
4. 出现提示时，确认删除。

16.6.3.2. 使用 certutil 删除证书

使用 certutil 从数据库中删除证书：

1. 打开实例的证书数据库目录。

```
/var/lib/pki/instance_name/alias
```

2. 使用 -L 选项运行 certutil，列出数据库中的证书。例如：

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

3. 通过使用 -D 选项运行 certutil 来删除证书。

```
certutil -D -d . -n certificate_nickname
```

例如：

```
certutil -D -d . -n "ServerCert cert-instance_name"
```

4. 再次列出证书，以确认证书已被删除。

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
```

有关使用 certutil 命令的详情请参考 http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

16.7. 更改 CA 证书的信任设置

CertificateCertificate Systemnbsp;System 子系统使用其证书数据库中的 CA 证书来验证在启用了 SSL 的通讯期间收到的证书。

可能需要更改存储在证书数据库的 CA 上的信任设置，或者永久更改。例如，如果访问或被破坏的证书存在问题，请将 CA 证书标记为不可信的实体，使用 CA 签名的证书向证书证书进行身份验证；系统.当

问题解决时，可以再次标记为受信任的 CA。

要永久取消信任 CA，请考虑将其证书从信任数据库中删除。具体说明请查看 第 16.6.3 节 “从数据库中删除证书”。

16.7.1. 通过控制台更改信任设置

要更改 CA 证书的信任设置，请执行以下操作：

1. 打开子系统控制台。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

2. 在 Configuration 选项卡中，来自 左侧导航树中的系统密钥和证书。

3. 选择 CA 证书 选项卡。

4. 选择要修改的 CA 证书，然后点 Edit。

5. 系统会打开一个提示信息，其中包含证书链(un)受信任的内容，确定您希望（不）信任它？

   点 yes 更改证书链的信任设置；按 no 保留原始的信任关系。

16.7.2. 使用 certutil 更改信任设置

要使用 certutil 更改证书的信任设置，请执行以下操作：

1. 打开实例的证书数据库目录。

```
cd /var/lib/pki/instance_name/alias
```

2. 使用 -L 选项运行 certutil，列出数据库中的证书。例如：

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

3. 通过使用 -M 选项运行 certutil 来更改证书的信任设置。

```
certutil -M -n cert_nickname -t trust -d .
```

例如：

```
certutil -M -n "Certificate Authority - Example Domain" -t TCu,TCu,TCu -d .
```

4. 再次列出证书，以确认证书信任已更改。

```
certutil -L -d .

Certificate Authority - Example Domain    CTu,CTu,CTu
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

有关使用 certutil 命令的详情请参考 http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

16.8. 管理由 子系统使用的令牌

CertificateCertificate Systemnbsp;System managers 两组令牌：子系统用于执行 PKI 任务和令牌通过子系统发布的令牌。这些管理任务专门引用由子系统使用的令牌。

有关管理智能卡令牌的详情，请参考 第 6 章 使用并配置令牌管理系统：TPS 和 TKS。

16.8.1. 检测令牌

要查看证书证书系统nbsp 是否可以检测到令牌，请使用 TokenInfo 实用程序。

```
TokenInfo /var/lib/pki/instance_name/alias
Database Path: /var/lib/pki/instance_name/alias
Found external module 'NSS Internal PKCS #11 Module'
```

这个实用程序会返回由 CertificateCertificate Systemnbsp;System 检测到的所有令牌，而不只恢复在证书证书系统空间中安装的令牌。

16.8.2. 查看令牌

要查看当前为 CertificateCertificate Systemnbsp;System 实例安装的令牌列表，请使用 modutil 工具。

1. 打开实例 别名 目录。例如：

```
cd /var/lib/pki/instance_name/alias
```

2. 显示有关已安装的 PKCS #11 模块的信息，以及使用 modutil 工具安装的对应令牌的信息。

```
modutil -dbdir . -nocertdb -list
```

16.8.3. 更改令牌的密码

存储子系统的密钥对和证书的令牌（加密）由密码保护（加密）。要解密密钥对或获得对其的访问，请输入令牌密码。当令牌首次访问令牌时，会设置此密码，通常是在 CertificateCertificate Systemnbsp;System 安装期间。

最好定期更改可保护服务器密钥和证书的密码。更改密码可最小化人发现密码的风险。要更改令牌的密码，请使用 certutil 命令行工具。

有关 certutil 的详情，请参考 http://www.mozilla.org/projects/security/pki/nss/tools/。

单点登录密码缓存在 password.conf 文件中存储令牌密码。每次更改令牌密码时，都必须手动更新此文件。有关通过 password.conf 文件管理密码的更多信息，请参阅 红帽认证系统规划、安装和部署指南。

第 17 章 在 RED HAT ENTERPRISE LINUX 7 中设置时间和日期

部分包含如何在 Red Hat Enterprise Linux 7 中设置时间和日期：

系统时间始终保存在协调的 通用基础镜像 (UTC)中，并在需要时将其转换为本地时间。本地时间 是您当前时区的实际时间，需要考虑 夏时时 (DST)。

timedatectl 程序作为 systemd 系统和服务管理器的一部分发布，可让您查看并更改系统时钟的配置。

更改当前时间

```
timedatectl set-time HH:MM:SS
```

将 HH 替换为一小时，MM 替换为一分钟，SS 替换为第二个，以双位形式键入。

更改当前日期

```
timedatectl set-time YYYY-MM-DD
```

将 YYYY 替换为一个四位年，MM 为两位数，使用月的两位日期替换 DD。

操作系统审计时间更改。如需更多信息，请参阅 Red Hat 证书系统规划、安装和部署指南中的 审计时间 更改事件部分。

Red Hat Certificate System 产品版本存储在 /usr/share/pki/CS_SERVER_VERSION 文件中。显示版本：

```
# cat /usr/share/pki/CS_SERVER_VERSION
Red Hat Certificate System 9.4 (Batch Update 3)
```

要查找正在运行的服务器的产品版本，请从浏览器中访问以下 URL：

- http://host_name:port_number/ca/admin/ca/getStatus
- http://host_name:port_number/kra/admin/kra/getStatus
- http://host_name:port_number/ocsp/admin/ocsp/getStatus
- http://host_name:port_number/tks/admin/tks/getStatus
- http://host_name:port_number/tps/admin/tps/getStatus

注意

请注意，每个组件都是一个单独的软件包，因此可能会有一个单独的版本号。以上将显示当前正在运行的每个组件的版本号。

第 19 章 更新红帽认证系统

要更新证书系统及其正在运行的操作系统，请使用 yum update 命令。下载、验证并安装证书系统和操作系统软件包更新。有关更新证书系统并验证更新是否成功的更多信息，请参阅《红帽证书系统 规划、安装和部署指南》中的更新证书系统软件包 章节。

第 20 章 故障排除

本章论述了在安装 CertificateCertificate Systemnbsp;System 时遇到的一些更常见的使用问题。

问：

初始化脚本返回 OK 状态，但 my CA 实例没有响应。这是因为什么？

答：

这不应发生。通常（但并不总是如此），这表示该 CA 的监听程序问题可能有很多不同的原因。检查 catalina.out、system，以及 调试 日志文件，以查看发生了哪些错误。这列出了几个常见错误。

个情况是 CA 的 PID，表示进程正在运行，但没有为服务器打开任何监听程序。这会在 catalina.out 文件中返回 Java 调用类错误：

```
Oct 29, 2010 4:15:44 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-9080
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:615)
        at org.apache.catalina.startup.Bootstrap.load(Bootstrap.java:243)
        at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:408)
Caused by: java.lang.UnsatisfiedLinkError: jss4
```

这可能意味着您有错误版本的 JSS 或 NSS。进程在路径中需要 libnss3.so。使用以下命令进行检查：

```
ldd /usr/lib64/libjss4.so
```

如果没有找到 libnss3.so，请尝试取消设置 LD_LIBRARY_PATH 变量并重启 CA。

```
unset LD_LIBRARY_PATH
systemctl restart pki-tomcatd@instance_name.service
```

问：

我无法打开 pkiconsole，并在 stdout 中看到 Java 例外。

答：

这可能意味着您安装了错误 JRE，或将错误 JRE 设置为默认值。运行 alternatives --config java 以查看选择什么 JRE。Red Hat Certificate Systemnbsp;Hat Certificate Red Hat

---

问：

我试图运行 pkiconsole，并在 stdout 中获取套接字例外。这是因为什么？

答：

这意味着存在端口问题。管理端口有不正确的 SSL 设置（表示 server.xml中存在错误的配置），或者为访问 admin 接口提供了错误的端口。

端口错误类似如下：

```
NSS Cipher Supported '0xff04'
java.io.IOException: SocketException cannot read on socket
        at org.mozilla.jss.ssl.SSLSocket.read(SSLSocket.java:1006)
        at org.mozilla.jss.ssl.SSLInputStream.read(SSLInputStream.java:70)
        at
com.netscape.admin.certsrv.misc.HttpInputStream.fill(HttpInputStream.java:303)
        at
com.netscape.admin.certsrv.misc.HttpInputStream.readLine(HttpInputStream.java:224)
        at
com.netscape.admin.certsrv.connection.JSSConnection.readHeader(JSSConnection.java:439)
        at
com.netscape.admin.certsrv.connection.JSSConnection.initReadResponse(JSSConnection.java:4
30)
        at
com.netscape.admin.certsrv.connection.JSSConnection.sendRequest(JSSConnection.java:344)

        at
com.netscape.admin.certsrv.connection.AdminConnection.processRequest(AdminConnection.java
:714)
        at
com.netscape.admin.certsrv.connection.AdminConnection.sendRequest(AdminConnection.java:62
3)
        at
com.netscape.admin.certsrv.connection.AdminConnection.sendRequest(AdminConnection.java:59
0)
        at
com.netscape.admin.certsrv.connection.AdminConnection.authType(AdminConnection.java:323)

        at
com.netscape.admin.certsrv.CMSServerInfo.getAuthType(CMSServerInfo.java:113)
        at com.netscape.admin.certsrv.CMSAdmin.run(CMSAdmin.java:499)
        at com.netscape.admin.certsrv.CMSAdmin.run(CMSAdmin.java:548)
        at com.netscape.admin.certsrv.Console.main(Console.java:1655)
```

问：

我试图注册证书，并且收到错误 "request is not submitted...Subject Name Not Found"？

答：

这通常使用自定义 LDAP 目录验证配置集，并显示目录操作失败。特别是，它会失败，因为它无法构造工作的 DN。该错误将位于 CA 的 调试日志 中。例如，这个配置集使用目录无法识别的自定义属性(MYATTRIBUTE)：

```
[14/Feb/2011:15:52:25][http-1244-Processor24]: BasicProfile: populate() policy
setid =userCertSet
[14/Feb/2011:15:52:25][http-1244-Processor24]: AuthTokenSubjectNameDefault:
populate start
[14/Feb/2011:15:52:25][http-1244-Processor24]: AuthTokenSubjectNameDefault:
java.io.IOException: Unknown AVA keyword 'MYATTRIBUTE'.
[14/Feb/2011:15:52:25][http-1244-Processor24]: ProfileSubmitServlet: populate
Subject Name Not Found
[14/Feb/2011:15:52:25][http-1244-Processor24]: CMSServlet: curDate=Mon Feb 14
15:52:25 PST 2011 id=caProfileSubmit time=13
```

任何自定义组件 - 属性、对象类和未注册的 OID - 在主体 DN 中使用都可能导致失败。在大多数情况下，RHC 2253 中定义的 X.509 属性应该用于主题 DN 而不是自定义属性。

问：

为什么没有发布注册的证书？

答：

这通常表示 CA 被错误配置。查找错误的主要位置是 debug 日志，它可以指示错误配置的位置。例如，这在映射程序中存在问题：

```
[31/Jul/2010:11:18:29][Thread-29]: LdapSimpleMap: cert subject
dn:UID=me,E=me@example.com,CN=yes
[31/Jul/2010:11:18:29][Thread-29]: Error mapping:
mapper=com.netscape.cms.publish.mappers.LdapSimpleMap@258fdcd0 error=Cannot
find a match in the LDAP server for certificate. netscape.ldap.LDAPException:
error result (32); matchedDN = ou=people,c=test; No such object
```

检查 CA 的 CS.cfg 文件中的 发布 配置，或者在 CA 控制台的 Publishing 选项卡中检查。在这个示例中，问题位于 mapping 参数中，它必须指向 现有的 LDAP 后缀：

```
ca.publish.mapper.instance.LdapUserCertMap.dnPattern=UID=$subj.UID,dc=publish
```

问：

如何从远程主机打开 pkiconsole 工具？

答：

在某些情况下，管理员希望从远程主机在证书系统服务器上打开 pkiconsole。为此，管理员可以使用虚拟网络计算(VNC)连接：

1. 在 Red Hat Certificate System 服务器中设置 VNC 服务器。

重要

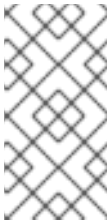
pkiconsole 工具无法在启用了 Federal Information Processing Standard(FIPS)模式的服务器中运行。如果证书系统服务器上启用了 FIPS 模式，请在 Red Hat Enterprise Linux 中使用不同的主机来运行 VNC 服务器。

有关安装 VNC 服务器的详情，请查看 Red Hat System Administrator 指南中的 VNC 服务器 部分。

2. 使用 VNC viewer 连接到运行 VNC 服务器的主机。详情请查看 Red Hat System Administrator 指南中的 VNC Viewer 部分。

3. 在 VNC 窗口中打开 pkiconsole 工具。例如：

```
# pkiconsole https://server.example.com:8443/ca
```



注意

VNC viewer 可用于不同类型的操作系统。但是，红帽只支持从集成软件仓库在 Red Hat Enterprise Linux 中安装 VNC viewer。

问：

当 LDAP 服务器没有响应时，我该怎么办？

答：

如果用于内部数据库的 Red Hat Directory Server 实例没有运行，则发生连接问题或 TLS 连接失败，那么您将无法连接到依赖它的子系统实例。实例调试日志将专门识别 LDAP 连接的问题。例如，如果 LDAP 服务器没有在线：

```
[02/Apr/2019:15:55:41][authorityMonitor]: authorityMonitor: failed to get LDAPConnection.
Retrying in 1 second.
[02/Apr/2019:15:55:42][authorityMonitor]: In LdapBoundConnFactory::getConn()
[02/Apr/2019:15:55:42][authorityMonitor]: masterConn is null.
[02/Apr/2019:15:55:42][authorityMonitor]: makeConnection: errorIfDown true
[02/Apr/2019:15:55:42][authorityMonitor]: TCP Keep-Alive: true
java.net.ConnectException: Connection refused (Connection refused)
```

```
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
    at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
[02/Apr/2019:15:55:42][authorityMonitor]: Can't create master connection in
LdapBoundConnFactory::getConn!
    Could not connect to LDAP server host example911.redhat.com port 389 Error
netscape.ldap.LDAPException:
        Unable to create socket: java.net.ConnectException: Connection refused (Connection
refused) (-1)
```

在修复底层网络问题后，比如电缆未插入后，Red Hat Directory Server 被停止、发生大量数据包丢失，或者确保可以重新创建 TLS 连接，然后遇到问题并启动证书系统实例：

```
# systemctl stop pki-tomcatd-nuxwdog@instance_name.service
```

```
# systemctl start pki-tomcatd-nuxwdog@instance_name.service
```

第 21 章 子系统控制和维护

本章提供了有关如何控制（启动、停止、重启和状态检查）红帽证书系统子系统以及常规维护（健康检查）建议的信息。

21.1. 启动、停止、重启和 OBTAINING 状态

Red Hat Certificate System 子系统实例可以使用 Red Hat Enterprise Linux 7 上的 systemctl 程序停止并启动。

启动实例：

```
# systemctl start unit_file@instance_name.service
```

停止实例：

```
# systemctl stop unit_file@instance_name.service
```

重启实例：

```
# systemctl restart unit_file@instance_name.service
```

显示实例状态：

```
# systemctl status unit_file@instance_name.service
```

unit_file 具有以下值之一：

- pki-tomcat: 在禁用 watchdog 时
- pki-tomcat-nuxwdog: With enabled watchdog

21.2. 子系统健康检查

管理员定期监控可能的故障很重要，例如：

- 由完整磁盘导致的审计故障
- 由 HSM 连接问题造成的签名失败
- LDAP 服务器连接问题
- 以此类推

您可以按照 第 13.9 节 “运行自助测试” 中描述的需求运行自我证明。

部分 V. 参考信息

附录 A. 证书配置集输入和输出参考

配置集输入和输出在证书请求中定义预期的输入参数，以及注册结果的输出格式。与 Red Hat Certificate Systemnbsp 中的很多其他组件一样；红帽证书证书系统nbsp；System，配置集输入和输出作为 JAVA 插件实现，以提供自定义和灵活性。本附录提供了默认输入和输出插件的参考。

- 第 A.1 节 “输入参考”
- 第 A.2 节 “输出参考”

A.1. 输入参考

输入会在与特定证书配置集关联的注册页面中放置某些字段。为证书配置集设置的输入用于通过适当的字段动态生成注册页面。这些输入字段会为配置集收集必要信息来生成最终证书。

A.1.1. 证书请求输入

证书请求输入用于将证书请求粘贴到注册表中的注册。它允许从下拉列表中设置请求格式，并提供输入字段来粘贴请求。

这个输入将以下字段放在注册表单中：

- 证书请求类型.此下拉菜单允许用户指定证书请求类型。选择为 PKCS #10 或 CRMF。PKCS #10 和 CRMF 支持通过 Cryptographic 消息语法(CMC)注册的证书管理消息。
- 证书请求.这是用于粘贴请求的文本区域。

例 A.1.

```
caAdminCert.cfg:input.i1.class_id=certReqInputImpl
```

A.1.2. CMC 证书请求输入

CMC 证书请求输入使用证书消息 over CMS(CMC)证书请求以请求形式提交。请求类型必须是 PKCS #10 或 CRMF，唯一字段是需要粘贴请求的证书请求文本区域。

例 A.2.

```
caCMCUserCert.cfg:input.i1.class_id=cmcCertReqInputImpl
```

A.1.3. 双密钥生成输入

第二代输入用于生成双密钥对的注册，因此发布两个证书，一个用于签名，另一个用于加密。

这个输入将以下字段放入注册表单中：

- 密钥生成请求类型.此字段是一个只读字段，显示 **crmf** 作为请求类型。
- 密钥生成请求.此字段为加密和签名证书的密钥生成请求设置密钥大小选择。

例 A.3.

```
caDualCert.cfg:input.i1.class_id=dualKeyGenInputImpl
```

A.1.4. 文件增强输入

File-Signing 输入会设置用于签署文件的字段，以表明其未被篡改。

这个输入会创建以下字段：

- 密钥生成请求类型.此字段是一个只读字段，显示 **crmf** 作为请求类型。
- 密钥生成请求.这个输入会添加一个下拉菜单，选择要在密钥生成请求中使用的密钥大小。
- 未签署的文件 **URL**.这将指定要签名的文件的位置。

- 签署的文本.该操作会给出文件名。

例 A.4.

```
caAgentFileSigning.cfg:input.i2.class_id=fileSigningInputImpl
```

A.1.5. 镜像输入

Image 输入设置用于为镜像文件签名的字段。此输入创建的唯一字段是 Image URL，它提供了要签名的镜像的位置。

A.1.6. 密钥生成输入

密钥生成输入用于生成单个密钥对（通常是基于用户的证书注册）的注册。

这个输入将以下字段放入注册表单中：

- 密钥生成请求类型.此字段是一个只读字段，显示 crmf 作为请求类型。
- 密钥生成请求.这个输入会添加一个下拉菜单，选择要在密钥生成请求中使用的密钥大小。

例 A.5.

```
caDualCert.cfg:input.i1.class_id=keyGenInputImpl
```

A.1.7. nsHKeyCertRequest (Token Key) Input

Token Key 输入用于为硬件令牌注册密钥，以便代理稍后用于基于证书的身份验证。

这个输入将以下字段放入注册表单中：

- 令牌密钥 CUID.此字段为令牌设备提供 CUID（上下文的唯一用户 ID）。

- 令牌密钥用户公钥.此字段必须包含令牌用户的公钥。

例 A.6.

```
caTempTokenDeviceKeyEnrollment.cfg:input.i1.class_id=nsHKeyCertReqInputImpl
```

A.1.8. nsNKeyCertRequest (Token User Key) Input

Token User Key 输入用于为硬件令牌的用户注册密钥，以便代理以后使用令牌进行基于证书的身份验证。这个输入将以下字段放入注册表单中：

- 令牌密钥用户 UID.此字段提供令牌设备的 LDAP 条目的 UID。
- 令牌密钥用户公钥.此字段必须包含令牌用户的公钥。

例 A.7.

```
caTempTokenUserEncryptionKeyEnrollment.cfg:input.i1.class_id=nsNKeyCertReqInputImpl
```

A.1.9. 序列号续订输入

Serial Number Renewal Input 用于设置现有证书的序列号，以便 CA 可以拉取原始证书条目，并使用信息重新生成证书。输入将 Serial Number 字段插入到注册表单中。

这是唯一需要与续订表格搭配使用的输入；所有其他信息都由证书条目提供。

例 A.8.

```
caTokenUserEncryptionKeyRenewal.cfg:input.i1.class_id=serialNumRenewInputImpl
```

A.1.10. 主题 DN 输入

借助 Subject DN 输入，用户可以输入特定的 DN 作为证书主题名称，输入会将单个 Subject Name 字段插入到注册表单中。

-

例 A.9.

```
caAdminCert.cfg:input.i3.class_id=subjectDNInputImpl
```

A.1.11. 主题名称输入

当需要从用户收集 DN 参数时，可以使用 Subject Name 输入来注册。参数用于划分证书中的主题名称。这个输入将以下字段放入注册表单中：

- UID （LDAP 目录用户 ID）
- 电子邮件
- 通用名称 （用户名）
- 机构单元 （机构单元（应该）
- 机构（机构名称 ）
- 国家（用户所在国家 /地区）

例 A.10.

```
caDualCert.cfg:input.i2.class_id=subjectNameInputImpl
```

A.1.12. 提交信息输入

提交者信息输入收集证书请求者的信息，如姓名、电子邮件和电话。

这个输入将以下字段放入注册表单中：

- requester Name

- *请求者电子邮件*

- ***requester Phone***

例 A.11.

```
caAdminCert.cfg:input.i2.class_id=submitterInfoInputImpl
```

A.1.13. 通用输入

*借助通用输入，管理员可以指定要与处理特征的扩展插件搭配使用的任意数量的输入字段。例如，在模式的 **Subject Alternative Name Extension Default** 插件中使用 **ccm** 和 **GUID** 参数：*

例 A.12.

```
input.i3.class_id=genericInputImpl
input.i3.params.gi_display_name0=ccm
input.i3.params.gi_param_enable0=true
input.i3.params.gi_param_name0=ccm
input.i3.params.gi_display_name1=GUID
input.i3.params.gi_param_enable1=true
input.i3.params.gi_param_name1=GUID
input.i3.params.gi_num=2
…
policyset.set1.p6.default.class_id=subjectAltNameExtDefaultImpl
policyset.set1.p6.default.name=Subject Alternative Name Extension Default
policyset.set1.p6.default.params.subjAltExtGNEnable_0=true
policyset.set1.p6.default.params.subjAltExtGNEnable_1=true
policyset.set1.p6.default.params.subjAltExtPattern_0=$request.ccm$
policyset.set1.p6.default.params.subjAltExtType_0=DNSName
policyset.set1.p6.default.params.subjAltExtPattern_1=
(Any)1.3.6.1.4.1.311.25.1,0410$request.GUID$
policyset.set1.p6.default.params.subjAltExtType_1=OtherName
policyset.set1.p6.default.params.subjAltNameExtCritical=false
policyset.set1.p6.default.params.subjAltNameNumGNs=2
```

A.1.14. 主题备用名称扩展输入

*主题备用名称扩展输入与主题备用名称扩展默认插件一起使用。它允许管理员启用 **URI** 中的编号参数，其格式为 **req_san_pattern_#** 进入输入，因此 **SubjectAltNameExt** 扩展。例如，包含以下内容的 **URI**：*

```
...&req_san_pattern_0=host0.Example.com&req_san_pattern_1=host1.Example.com
```

-

将 host0.Example.com 和 host1.Example.com 注入以下配置集中的 SubjectAltNameExt 扩展。

例 A.13.

```
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl
policyset.serverCertSet.9.constraint.class_id=noConstraintImpl
policyset.serverCertSet.9.constraint.name=No Constraint
policyset.serverCertSet.9.default.class_id=subjectAltNameExtDefaultImpl
policyset.serverCertSet.9.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_0=$request.req_san_pattern_0$
policyset.serverCertSet.9.default.params.subjAltExtType_0=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.req_san_pattern_1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=$request.req_san_pattern_2$
policyset.serverCertSet.9.default.params.subjAltExtType_2=DNSName
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=2
```

A.2. 输出参考

输出是对成功注册的最终用户的响应。

A.2.1. 证书输出

此输出显示证书为 prettyprint。此输出无法配置或更改。它不以广泛打印格式显示证书以外的任何内容。

对于任何自动化注册，需要指定此输出。当用户使用自动化注册方法成功验证后，证书会自动生成，此输出页面将返回到用户。在代理批准的注册中，用户可能会在签发后获得证书，方法是在最终用户页面中提供请求 ID。

例 A.14.

```
caAdminCert.cfg:output.o1.class_id=certOutputImpl
```

A.2.2. PKCS #7 Output

此输出返回 PKCS #7 格式的证书和证书链。PKCS #7 格式是 Cryptographic Message 语法标准，用于签名。此输出无法配置或更改。

例 A.15.

```
caAgentFileSigning.cfg:output.o1.class_id=pkcs7OutputImpl
```

A.2.3. nsNSKeyOutput

此类实现输出插件，该插件返回用于令牌密钥的 DER 编码证书。

A.2.4. CMMF 输出

此输出返回证书管理消息格式(CMMF)。CMMF 监管 PKI 不同部分之间的通信，用于请求证书并请求证书撤销。

附录 B. 证书和 CRL 的默认、限制和扩展

本附录解释了 X.509 v3 定义的标准证书扩展，以及在 X.509 v3 之前发布的产品版本中使用的 Netscape 定义的扩展。它提供了扩展用于特定类型证书的建议，包括 PKIX 第 1 部分建议。

重要

本附录是对使用的默认、约束和 CRL 扩展的引用，这些扩展在 Red Hat Certificate Systemnbsp 中被使用或被配置 ;Hat Certificate Red Hat Certificate Systemnbsp;System.有关证书和 CRL 扩展的完整参考和说明，请参阅 RFC 3280。

本附录包含以下部分：

- 第 B.1 节 “默认参考”
- 第 B.2 节 “约束参考”
- 第 B.3 节 “标准 X.509 v3 证书扩展参考”
- 第 B.4 节 “CRL 扩展”

B.1. 默认参考

默认值用于定义证书的内容。本节列出了并定义预定义的默认值。

B.1.1. 授权信息扩展默认

此默认附加授权信息访问扩展。此扩展指定应用程序验证证书如何访问信息，如在线验证服务和 CA 策略数据，以及签发证书的 CA。此扩展不应用于直接指向 CA 维护的 CRL 位置 ; CRL Distribution Points 扩展 第 B.1.7 节 “CRL Distribution Points Extension Default” 提供了有关 CRL 位置的引用。

有关这个扩展的常规信息，请参阅 第 B.3.1 节 “authorityInfoAccess”。

以下限制可使用此默认值定义：

- *扩展约束；请参阅 第 B.2.4 节 “扩展约束”。*
- *无限制; 请参阅 第 B.2.8 节 “No Constraint”。*

此默认值最多可定义五个位置，每个位置的参数。在表格中使用 n 标记，以显示参数所关联的位置。

表 B.1. authority Info Access Extension 默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
Method_n	指定检索在 中出现扩展所发布证书的 CA 的其他信息的访问方法。这是以下值之一： • ocsp (1.3.6.1.5.5.7.48.1). • calssuers (1.3.6.1.5.5.7.48.2) • renewal (2.16.840.1.113730.16.1)

参数	描述
LocationType_n	指定包含签发证书的 CA 的其他信息的位置的常规名称类型。这是以下类型之一： • DirectoryName • DNSName • EDIPartyName • ipaddress • OID • RFC822Name • URIName
Location_n	指定地址或位置以获取签发证书的 CA 的额外信息。 • 对于 directoryName，该值必须是 X.500 名称的字符串，类似于证书中的主题名称。例如, cn=SubCA, ou=Research Dept, o=Example Corporation, c=US. • 对于 dNSName，该值必须是有

参数	描述
	效的完全限定域名。例如，testCA.example.com。 • 对于 EDI 第三方名称，该值必须是 IA5String。例如，Sartor 公司示例。 • 对于 iPAddress，该值必须是有效的 IP 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • 对于 OID，该值必须是唯一且有效的 OID，用点分隔的数值组件表示法指定。例如，1.2.3.4.55.6.5.99。 • 对于 RFC822Name，该值必须是有效的互联网电子邮件地址。 • 对于 URIName，该值必须是基于 URL 语法和编码规则的非参考资源标识符(URI)。名称必须同时包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如 ：http://ocspResponder.example.com:8000CertificateCertificate Systemnbsp;System 允许 IPv4 和 IPv6 IP 地址。
Enable_n	指定是否启用此位置。选择 true 将其标记为 set；选择 false 来禁用它。

B.1.2. 颁发机构键标识符默认值

此默认将授权密钥标识符扩展附加到证书。扩展标识与 CA 用来签署证书的私钥对应的公钥。这个默认没有参数。如果使用，这个扩展会包含在证书中，并包含公钥信息。

此默认值采用以下约束：

- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

有关这个扩展的常规信息，请参阅 第 B.3.2 节 “authorityKeyIdentifier”。

B.1.3. 身份验证令牌对象名称默认

此配置集默认根据身份验证令牌(AuthToken)对象中的属性值填充 subject 名称。

此默认插件可用于基于目录的身份验证管理器。基于 Directory 的用户 Dual-Use Certificate Enrollment 证书配置集有两个输入参数 UID 和密码。基于目录的身份验证管理器检查给定的 UID 和密码是否正确。

此外，基于目录的身份验证管理器可识别出发布证书的主题名称。它通过使用 AuthToken 中的用户的 DN 值来形成主题名称。

此默认值负责从 AuthToken 读取主体名称并将其放在证书请求中，以便最终证书包含主题名称。

以下限制可使用此默认值定义：

- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.4. 基本限制扩展默认值

此默认向证书附加 Basic Constraint 扩展。扩展标识证书管理器是否是一个 CA。该扩展也会在证书链验证过程中使用，以识别 CA 证书并应用证书链路径长度限制。

有关这个扩展的常规信息，请参阅 第 B.3.3 节 “basicConstraints”。

以下限制可使用此默认值定义：

- 基本限制扩展约束；请参阅 第 B.2.1 节 “基本限制扩展约束”。
- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制；请参阅 第 B.2.8 节 “No Constraint”。

表 B.2. 基本限制默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键；选择 false 来标记非关键。
IsCA	指定证书主题是否为 CA。使用 true 时，服务器会检查 PathLen 参数，并在证书中设置指定的路径长度。使用 false 时，服务器会将证书主题视为非 CA，并忽略为 PathLen 参数指定的值。

参数	描述
PathLen	指定路径长度，在下面链接的最大 CA 证书数（从属至）发出的从属 CA 证书数。路径长度会影响在证书验证过程中使用的 CA 证书的数量。链从验证和移动的最终证书开始。 如果以长期证书设置了扩展，则 maxPathLen 参数无效。 允许的可能值为 0 或 n。该值应该小于 CA 签名证书的 Basic Constraints 扩展中指定的路径长度。0 指定在从属 CA 证书下不允许从属 CA 证书；在路径中只能跟踪终止证书。n 必须是大于零的整数。它指定从属 CA 证书下允许的最大下级 CA 证书数。 如果该字段为空，则路径长度默认为由签发者证书中的 Basic Constraints 扩展中设置的路径长度确定的值。如果签发者的路径长度不受限制，则从属 CA 证书中的路径长度也会无限。如果签发者的路径长度大于零，则从属 CA 证书中的路径长度将设置为比签发者的路径长度小的值；例如，如果签发者的路径长度为 4，则子协调 CA 证书中的路径长度将设置为 3。

B.1.5. CA 有效期默认值

这个默认在 CA 证书注册或续订配置集中添加选项以绕过 CA 的签名证书的过期约束。这意味着签发的 CA 证书可到期日期超过发出 CA 签名证书过期日期。

以下限制可使用此默认值定义：

- 有效期约束；请参阅 第 B.2.14 节 “有效期约束”。
- 无限制；请参阅 第 B.2.8 节 “No Constraint”。

表 B.3. CA Validity Default 参数

参数	描述
bypassCAnotafterrange	为请求 CA 是否可以请求一个有效周期超过发出 CA 有效期期的证书设置默认值。
range	指定此证书的绝对有效期期，以天为单位。
startTime	根据当前时间，设置有效期时。

B.1.6. 证书策略扩展默认值

此默认将证书策略映射扩展附加到证书模板中。此扩展定义一个或多个策略，指示签发证书的策略以及可以使用该证书的目的。此默认值定义最多五个策略，但这个值可以被更改。

有关这个扩展的常规信息，请查看 第 B.3.4 节 “certificatePoliciesExt”

表 B.4. 证书策略扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键；选择 false 来标记非关键。
numCertPolicies	指定可定义的策略数量。默认值为 5。
enable	选择 true 以启用策略；选择 false 来禁用策略。
policyId	指定策略的 OID 标识符。

参数	描述
cpsURI.enable	扩展可以包括 URI 到签发者的证书声明。选择 true 以启用 URI；选择 false 来禁用 URI。
CPSURI.value	此值是对 CA 发布的认证实践声明(CPS)的指针。指针采用 URI 的形式。
usernotice.enable	扩展可以包括 URI 到签发者的证书声明，或者可以嵌入签发者信息，如用户以文本形式通知。选择 true 来启用用户通知；选择 false 来禁用用户通知。
usernotice.noticeReference.noticeNumbers	此可选用户 notice 参数是一系列数字，指向其他位置存储的消息。
usernotice.noticeReference.organization	此可选的 user notice 参数指定公司的名称。
usernotice.explicitText.value	此可选用户 notice 参数包含证书内的消息。

B.1.7. CRL Distribution Points Extension Default

此默认将 CRL Distribution Points 扩展附加到证书。此扩展标识验证证书的应用程序可以获取 CRL 信息来验证证书的撤销状态的位置。

有关这个扩展的常规信息，请参阅 第 B.3.5 节 “CRLDistributionPoints”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

此默认定义最多五个位置，其中包含每个位置的参数。在表格中使用 n 标记，以显示参数所关联的位置。

表 B.5. CRL Distribution Points Extension 配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
Type_n	指定 CRL 分发点的类型。Permissible 值为 DirectoryName、URIName 或 RelativeToIssuer。这个类型必须与 Name 字段中的值对应。
Name_n	指定 CRL 分发点的名称，名称可以是以下格式： • RFC 2253 语法中的 X.500 目录名称。名称与证书中的主题名称类似，如 cn=CA Central、ou=Research Dept、o=Example Corporation、c=US。 • A URIName, such as http://testCA.example.com:80. • 指定相对于 CRL 签发者的位置的 RDN。在本例中，Type 属性的值必须是 RelativeToIssuer。

参数	描述
Reasons_n	指定在分发点上维护的 CRL 所涉及到的撤销原因。提供以下常量的逗号分隔列表： • 未使用 • keyCompromise • cACompromise • affiliationChanged • 取代 • cessationOfOperation • certificateHold

参数	描述
IssuerType_n	*指定在分发点上签名了 CRL 的签发者的命名类型。签发者名称可采用以下格式：* • *RFC822Name* • *DirectoryName* • *DNSName* • *EDIPartyName* • *URIName* • *ipaddress* • *OIDName* • *OtherName*
IssuerName_n	*指定为 CRL 签名的 CRL 签发者的名称格式。允许的允许的值如下：* •

参数	描述
	对于 RFC822Name，该值必须是有效的互联网电子邮件地址。例如 ：testCA@example.com • 对于 DirectoryName，该值必须是 X.500 名称的字符串，类似于证书中的主题名称。例如, cn=SubCA, ou=Research Dept, o=Example Corporation, c=US. • 对于 DNSName，该值必须是有效的完全限定域名。例如，testCA.example.com。 • 对于 EDI 第三方名称，该值必须是 IA5String。例如，Sartor 公司示例。 • 对于 URIName，该值必须是采用 URL 语法和编码规则的非相对 URI。名称必须同时包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如 ：http://testCA.example.comCertificate Certificate Systemnbsp;System 支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，值必须是有效的 IP 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • 对于 OIDName，该值必须是唯一且有效的 OID，用点分隔的数值组件表示法指定。例如，1.2.3.4.55.6.5.99。

参数	描述
	OtherName 用于其他格式的名称，它支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。 OtherName 的格式 （类型）oid，字符串。例如，(IA5String)1.2.3.4,MyExample. 此参数的值必须与 issuerName 字段中的值对应。

B.1.8. 扩展密钥使用扩展默认值

此默认向证书附加扩展密钥使用扩展。

有关这个扩展的常规信息，请参阅 第 B.3.6 节 “extKeyUsage”。

除使用密钥使用扩展中指示的基本目的外，扩展还可识别这些目的。例如，如果密钥使用扩展标识了签名密钥，则扩展密钥使用扩展可以缩小密钥只用于签名 OCSP 响应或只有 Java™ 小程序。

表 B.6. 扩展密钥使用方法的 PKIX 使用量定义

使用	OID
服务器身份验证	1.3.6.1.5.5.7.3.1
客户端身份验证	1.3.6.1.5.5.7.3.2
代码签名	1.3.6.1.5.5.7.3.3
电子邮件	1.3.6.1.5.5.7.3.4

使用	OID
IPsec 结束系统	1.3.6.1.5.5.7.3.5
IPsec 隧道	1.3.6.1.5.5.7.3.6
IPsec 用户	1.3.6.1.5.5.7.3.7
时间戳	1.3.6.1.5.5.7.3.8

Windows 2000 可以加密硬盘中的文件，称为加密文件系统(EFS)，使用包含扩展密钥使用扩展密钥扩展以及以下两个 OID 的证书：

1.3.6.1.4.1.311.10.3.4（EFS 证书）

1.3.6.1.4.1.311.10.3.4.1（EFS 恢复证书）

当用户丢失私钥以及需要使用该密钥加密的数据时，恢复代理使用 EFS 恢复证书。CertificateCertificate Systemnbsp;System 支持这两个 OID，并允许发布包含这些 OID 的扩展密钥使用扩展的证书。

普通用户证书应该只使用 EFS OID 而不是恢复 OID 创建。

以下限制可使用此默认值定义：

- 扩展密钥使用约束；请参阅 第 B.2.3 节 “扩展密钥使用扩展约束”。
- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制；请参阅 第 B.2.8 节 “No Constraint”。

表 B.7. 扩展密钥使用扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
OID	指定用于标识密钥使用用途的 OID。Permissible 值是一个唯一且有效的 OID，在点分隔的数字组件标记中指定。例如，2.16.840.1.113730.1.99。根据密钥使用目的，OID 可以被 PKIX（在 表 B.6 “扩展密钥使用方法的 PKIX 使用量定义”中列出）或自定义 OID 来指定。自定义 OID 必须位于公司使用的 ID 已注册子树中。虽然可以使用自定义 OID 评估和测试证书证书证书证书nbsp;System，在生产环境中，遵循用于定义 OID 和注册 ID 的 ISO 规则，但可以使用自定义 OID 进行评估和测试证书证书。

B.1.9. 最新的 CRL 扩展默认值

此默认向证书附加 Freshest CRL 扩展。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

此默认值定义了五个位置，其中包含每个位置的参数。在表格中使用 n 标记，以显示参数所关联的位置。

表 B.8. 最新的 CRL 扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
PointEnable_n	选择 true 即可启用此点；选择 false 来禁用此点。

参数	描述
PointType_n	指定发布点的类型，可以是 DirectoryName 或 URIName。
PointName_n	• 如果将 pointType 设置为 directoryName，则值必须是 X.500 名称，与证书中的主题名称类似。例如：cn=CACentral,ou=Research Dept,o=Example company,c=US。 • 如果将 pointType 设置为 URIName，则名称必须是 URI，即指定主机的绝对路径名称。例如 ：http://testCA.example.com/get/crls/here/
PointIssuerName_n	指定已签署 CRL 的签发者的名称。名称可采用以下格式： • 对于 RFC822Name，该值必须是有效的互联网电子邮件地址。例如 ：testCA@example.com • 对于 DirectoryName，该值必须是 X.500 名称的字符串，类似于证书中的主题名称。例如, cn=SubCA, ou=Research Dept, o=Example Corporation, c=US. • 对于 DNSName，该值必须是有效的完全限定域名。例如，testCA.example.com。

参数	描述
	• 对于 EDI 第三方名称，该值必须是 IA5String。例如，Sartor 公司示例。 • 对于 URIName，该值必须是采用 URL 语法和编码规则的非相对 URI。名称必须同时包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如 ：http://testCA.example.comCertificate Certificate Systemnbsp;System 支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，值必须是有效的 IP 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • 对于 OIDName，该值必须是唯一且有效的 OID，用点分隔的数值组件表示法指定。例如，1.2.3.4.55.6.5.99。 • OtherName 用于其他格式的名称，它支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。 OtherName 的格式 （类型）oid，字符串。例如，(IA5String)1.2.3.4,MyExample.

参数	描述
	name 值必须符合 PointType_ 中指定的格式。
PointType_n	指定为 CRL 签名的 CRL 签发者的一般名称类型。允许的允许的值如下： • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • ipaddress • OIDName • OtherName 此参数的值必须与 PointIssuerName 字段的值对应。

B.1.10. 通用扩展默认

此扩展允许使用用户决定的数据创建通用扩展。默认确保正确填充通用扩展。

表 B.9. 通用扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
genericExtOID	指定 extensions OID 标识符。
genericExtData	扩展中包含的二进制数据。

B.1.11. 禁止任何策略扩展默认值

禁止任何策略扩展可用于向 CA 发布的证书。禁止 any-policy 表示特殊的 anyPolicy OID（值为 { 2 5 29 32 0 }）并不被视为其他证书策略的显式匹配项。

表 B.10. 禁止任何策略扩展默认配置参数

参数	描述
Critical	这个策略必须标记为 critical。选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
SkipCerts	此参数指定在不再允许任何策略前可能会出现在路径中的额外证书数量。值 1 表示，任何策略可能会被处理在此证书上发布的证书，但不能处理路径中的额外证书。

B.1.12. 签发者备用名称扩展默认值

这个默认会向证书附加签发者替代名称扩展。Issuer Alternative Name 扩展用于将互联网风格的身份与证书签发者关联。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

此默认值定义了五个位置，其中包含每个位置的参数。在表格中使用 n 标记，以显示参数所关联的位置。

表 B.11. 签发者备用名称扩展默认配置参数

参数	描述
Critical	选择 **true** 来标记这个扩展关键; 选择 **false** 来标记非关键。
issuerAltExtType	这将设置要使用的名称扩展类型，它可以是以下之一： • **RFC822Name** • **DirectoryName** • **DNSName** • **EDIPartyName** • **URIName** • **ipaddress** • **OIDName**

参数	描述
issuerAltExtPattern	指定要在扩展中包含的 request 属性值。属性值必须符合任何受支持的通用名称类型。permissible 值是证书请求中包含的 request 属性。 如果服务器在请求中找到 属性，它将在扩展中设置属性值，并为证书添加扩展。如果指定了多个属性，且请求中没有包括任何属性，服务器不会为证书添加 Issuer Alternative Name 扩展。如果不能从请求中使用合适的属性组成 issuerAlternativeName，则可以在不用任何令牌表达式的情况下使用字面字符串。例如，证书授权中心。

B.1.13. 主要使用扩展默认值

此默认将 Key Usage 扩展附加到证书。扩展指定使用证书中包含的密钥的目的，如数据签名、密钥加密或数据加密，其限制对使用密钥来预先确定的目的。

有关这个扩展的常规信息，请参阅 第 B.3.8 节 “keyUsage”。

以下限制可使用此默认值定义：

- 密钥使用约束；请参阅 第 B.2.6 节 “主要使用扩展约束”。
- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.12. 主要使用扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
digitalSignature	指定是否允许签名 SSL 客户端证书和 S/MIME 签名证书。选择 true 来设置。
nonRepudiation	指定是否将 用于 S/MIME 签名证书。选择 true 来设置。  **WARNING** 使用这个位是简介的。在为任何证书设置前，请仔细考虑其使用的合法后果。
keyEncipherment	指定主体中的公钥是否用于隔离私钥或 secret 密钥。这是为 SSL 服务器证书和 S/MIME 加密证书设置的。选择 true 来设置。
dataEncipherment	指定在使用主体的公钥而非密钥材料时，是否设置扩展。选择 true 来设置。
keyAgreement	指定每当将主题的公钥用于密钥协议时，是否设置扩展。选择 true 来设置。
keyCertsign	指定公钥是否用于验证其他证书的签名。此设置用于 CA 证书。选择 true 来设置 选项。
cRLSign	指定是否为为 CRL 签名的 CA 签名证书设置扩展。选择 true 来设置。
encipherOnly	指定当公钥只在执行密钥协议时加密数据时是否设置扩展。如果设置了这个位，则还应设置 keyAgreement。选择 true 来设置。

参数	描述
decipherOnly	指定当公钥只在执行密钥协议时解密数据时是否设置扩展。如果设置了这个位，则还应设置 **keyAgreement**。选择 **true** 来设置。

B.1.14. 名称限制扩展默认值

此默认为证书附加一个 **Name Constraints**扩展。扩展用于在 **CA** 证书中指明应位于证书链中主题名称或主题备用名称的命名空间。

有关这个扩展的常规信息，请参阅 第 B.3.9 节 “nameConstraints”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制；请参阅 第 B.2.8 节 “No Constraint”。

此默认值为允许的子树和排除子树定义最多五个位置，并为每个位置设置参数。在表格中使用 **n** 标记，以显示参数所关联的位置。

表 B.13. 名称限制默认配置参数

参数	描述
Critical	选择 **true** 来标记这个扩展关键；选择 **false** 来标记非关键。

参数	描述
PermittedSubtreesn.min	指定允许的子树的最小数量。 • -1 指定在扩展中不应设置该字段。 • 0 指定子树的最小数量为零。 • n 必须是大于零的整数。它设置所需的最小子树数。
PermittedSubtreesmax_n	指定允许的子树的最大数量。 • -1 指定在扩展中不应设置该字段。 • 0 指定子树的最大数量为零。 • n 必须是大于零的整数。它设置允许的最大子树数。

参数	描述
PermittedSubtreeNameChoice_n	***指定要在扩展中包含允许的子树的常规名称类型。允许的允许的值如下：*** • ***RFC822Name*** • ***DirectoryName*** • ***DNSName*** • ***EDIPartyName*** • ***URIName*** • ***ipaddress*** • ***OIDName*** • ***OtherName***
PermittedSubtreeNameValue_n	***指定要在扩展中包含允许的子树的常规名称值。*** • *对于* ***RFC822Name，该值必须***

参数	描述
	是有效的互联网电子邮件地址。例如 ：testCA@example.com • 对于 DirectoryName，该值必须是 X.500 名称的字符串，类似于证书中的主题名称。例如, cn=SubCA, ou=Research Dept, o=Example Corporation, c=US. • 对于 DNSName，该值必须是有效的完全限定域名。例如，testCA.example.com。 • 对于 EDI 第三方名称，该值必须是 IA5String。例如，Sartor 公司示例。 • 对于 URIName，该值必须是采用 URL 语法和编码规则的非相对 URI。名称必须同时包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如 ：http://testCA.example.comCertificate Certificate Systemnbsp;System 支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，该值必须是符合无类别域间路由(CIDR)标记的有效 IP 地址。IPv4 地址必须采用 n.n.n 格式，或带有子网掩码的 n.n.n.n /m - 例如 10.34.3.133 或 110.34.3.133/24。IPv6 地址还必须符合 CIDR 表示法；子网掩码的示例包括 2620:52:0:2203:527b:9dff:fe56:4495/4495/64 或 2001:db8::/64。 • 对于 OIDName，该值必须是唯一且有效的 OID，用点分隔的数值组件表示法指定。例如，1.2.3.4.55.6.5.99。 • OtherName 用于其他格式的名称，它支持 PrintableString、IA5String、UTF8String、BMPString、任何、和

参数	描述
	KerberosName。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。 OtherName 的格式 （类型）oid，字符串。例如，(IA5String)1.2.3.4,MyExample.
PermittedSubtreeEnable_n	选择 true 启用这个允许的子树条目。
ExcludedSubtreesn.min	指定排除子树的最小数量。 • -1 指定在扩展中不应设置该字段。 • 0 指定子树的最小数量为零。 • n 必须是大于零的整数。这将设置所需子树的最小数量。

参数	描述
ExcludedSubtreeMax_n	指定排除子树的最大数量。 • -1 指定在扩展中不应设置该字段。 • 0 指定子树的最大数量为零。 • n 必须是大于零的整数。这将设置允许的子树的最大数量。

参数	描述
ExcludedSubtreeNameChoice_n	指定要包含在扩展中的排除子树的常规名称类型。允许的允许的值如下： • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • ipaddress • OIDName • OtherName
ExcludedSubtreeNameValue_n	指定要在扩展中包含允许的子树的常规名称值。 • 对于 RFC822Name，该值必须是有效的互联网电子邮件地址。例如 ：testCA@example.com

参数	描述
	• 对于 DirectoryName，该值必须是 X.500 名称，与证书中的主题名称类似。例如, cn=SubCA, ou=Research Dept, o=Example Corporation, c=US. • 对于 DNSName，该值必须是有效的完全限定域名。例如，testCA.example.com。 • 对于 EDI 第三方名称，该值必须是 IA5String。例如，Sartor 公司示例。 • 对于 URIName，该值必须是采用 URL 语法和编码规则的非相对 URI。名称必须同时包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如 ：http://testCA.example.comCertificate Certificate Systemnbsp;System 支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，该值必须是符合无类别域间路由(CIDR)标记的有效 IP 地址。IPv4 地址必须采用 n.n.n 格式，或带有子网掩码的 n.n.n.n /m - 例如 10.34.3.133 或 110.34.3.133/24。IPv6 地址还必须符合 CIDR 表示法；子网掩码的示例包括 2620:52:0:2203:527b:9dff:fe56:4495/4495/64 或 2001:db8::/64。 • 对于 OIDName，该值必须是唯一且有效的 OID，用点分隔的数值组件表示法指定。例如，1.2.3.4.55.6.5.99。 • 对于 OtherName，值采用任何其他格式的名称。它支持 PrintableString、IA5String、UTF8String、BMPString、any、和 KerberosName。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。

参数	描述
	OtherName 的格式 （类型）oid，字符串。例如，(IA5String)1.2.3.4,MyExample.
ExcludedSubtreeEnable_n	选择 true 来启用此排除子树条目。

B.1.15. Netscape Certificate Type Extension Default

WARNING

这个扩展已过时。改为使用 Key Usage 或 Extended Key Usage 证书扩展。

此默认向证书附加 Netscape 证书类型扩展。扩展标识证书类型，如 CA 证书、服务器 SSL 证书、客户端 SSL 证书或 S/MIME 证书。这将限制使用证书来预先确定的目的。

B.1.16. Netscape Comment Extension 默认

WARNING

这个扩展已过时。

此默认向证书附加 Netscape Comment 扩展。该扩展可用于在证书中包含文本注释。能够解释注释的应用程序在证书被使用或查看时显示它。

有关这个扩展的常规信息，请参阅 第 B.4.3.2 节 “netscape-comment”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.14. Netscape Comment Extension 配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
CommentContent	指定要在证书中出现的注释的内容。

B.1.17. 没有默认扩展

如果没有使用默认值，可以使用这个默认值来设置限制。这个默认值没有设置，且没有默认值，但允许设置所有限制。

B.1.18. OCSP No Check Extension Default

这会默认为证书附加 OCSP No Check 扩展。只在 OCSP 响应器证书中使用扩展，指示与 OCSP 兼容的应用程序如何检查证书的撤销状态，并使用授权 OCSP 响应为 OCSP 响应签名。

有关这个扩展的常规信息，请参阅 第 B.3.10 节 “OCSPNocheck”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.15. OCSP No Check Extension 默认配置参数

参数	描述

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。

B.1.19. 策略限制扩展默认值

此默认值为证书附加一个策略限制扩展。扩展仅可用于 CA 证书，以两种方式限制路径验证：要禁止策略映射，或者要求路径中的每个证书都包含可接受的策略标识符。默认值可以同时指定 ReqExplicitPolicy 和 InhibitPolicyMapping。PKIX 标准要求，如果证书中存在，则扩展不能由 null 序列组成。必须至少存在两个指定字段之一。

有关这个扩展的常规信息，请参阅 第 B.3.11 节 “policyConstraints”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.16. 策略限制默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。

参数	描述
reqExplicitPolicy	指定在需要显式策略前路径中允许的证书总数。这是在需要可接受的策略前，可串联在下级 CA 证书下的 CA 证书的数量。 • -1 指定在扩展中不应设置该字段。 • 0 指定在需要显式策略前，路径中没有允许下级 CA 证书。 • n 必须是大于零的整数。它指定路径中需要显式策略前允许的最大下级 CA 证书数。 这个数字会影响在证书验证过程中要使用的 CA 证书数量。链从验证和移动链的最终证书开始。如果以长期证书设置了扩展，则参数无效。

参数	描述
inhibitPolicyMapping	在不再允许策略映射前，指定路径中允许的证书总数。 • -1 指定在扩展中不应设置该字段。 • 0 指定在不再允许策略映射前，路径中不允许从属 CA 证书。 • n 必须是大于零的整数。它指定在不再允许策略映射前路径允许的最大下级 CA 证书数。例如，1 表示可在此证书的主题发布的证书中处理策略映射，但不在路径中的额外证书中处理。

B.1.20. 策略映射程序扩展默认

此默认向证书附加 Policy Mappings 扩展。扩展列出了 OID 对，每个对标识两个 CA 的策略语句。对表示一个 CA 的对应策略等同于另一个 CA 的策略。在跨认证上下文中，扩展可能很有用。如果支持，扩展仅包含在 CA 证书中。默认将一个 CA 的策略语句映射到另一个 CA，方法是对分配给其策略声明的 OID 进行匹配

每个对都由两个参数： issuerDomainPolicy 和 subjectDomainPolicy 定义。对称发出 CA 会考虑与主题 CA 的 subjectDomainPolicy 等效的 issuerDomainPolicy。发出 CA 的用户可能会接受特定应用程序的 issuerDomainPolicy。策略映射告知这些用户与主题 CA 关联的策略等同于其接受的策略。

有关这个扩展的常规信息，请参阅 第 B.3.12 节 “policyMappings”。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.17. 策略映射默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键; 选择 false 来标记非关键。
IssuerDomainPolicy_n	指定分配给发布 CA 的 policy 语句的 OID，以便与另一个 CA 的 policy 语句映射。例如，1.2.3.4.5。
SubjectDomainPolicy_n	指定分配给发布 CA 的主题 CA 的 policy 语句的 OID，该语句与发出 CA 的 policy 语句对应。例如：6.7.8.9.10。

B.1.21. 私钥使用周期扩展默认值

Private Key Usage Period 扩展允许证书签发者为私钥指定不同于证书本身的有效周期。这个扩展用于数字签名密钥。

表 B.18. 私钥使用定期配置参数

参数	描述
Critical	这个扩展应该总是非关键。
puStartTime	此参数设置开始时间。默认值为 0， 它从激活扩展时开始有效周期。
puDurationDays	此参数设置用量周期的时间。默认值为 365， 它从激活扩展时将有效期期设置为 365 天。

B.1.22. 签名算法

此默认在证书请求中附加签名算法。此默认向代理提供可能的算法， 可用于签署证书。

以下限制可使用此默认值定义：

- 签名算法; 请参阅 第 B.2.10 节 “签名算法”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.19. 签名算法默认配置参数

参数	描述
signingAlg	指定要用于创建此证书的默认签名算法。代理可通过指定 签名AlgsAllowed 参数中包含的值之一来覆盖这个值。
signingAlgsAllowed	指定可用于为此证书签名的签名算法。这些算法可以是以下任意一种或全部算法： • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA

B.1.23. 主题名称扩展默认值

此默认向证书附加 Subject Alternative Name 扩展。扩展将其他身份（如电子邮件地址、DNS 名称、IP 地址（包括 IPv4 和 IPv6）绑定到证书的主题。标准要求如果证书主题字段包含空序列，那么主题备用名称扩展必须包含主题的替代名称，并且扩展名标记为关键。

对于任何基于目录的身份验证方法，证书Certificate Systemnbsp；System 可以检索任何字符串和字节属性的值，并在证书请求中设置它们。这些属性通过在自动注册模块中定义的 ldapStringAttributes

和 IdapByteAttributes 字段中输入它们进行设置。

如果验证的属性 - 存储在 LDAP 数据库中的属性需要是该扩展的一部分，则使用 $request.X$ token 的值。

另外有一个额外的属性，可将通用唯一标识符(UUID)插入到 subject alt name 中。这个选项为版本 4 UUID 生成随机数字；通过引用服务器来定义该模式，该服务器将在额外的 subjAltExtSource 参数中生成数字。

示例中配置了基本的主题备用名称扩展默认设置。

例 B.1. 默认主题备用名称扩展配置

```
policyset.serverCertSet.9.constraint.name=No Constraint
policyset.serverCertSet.9.default.class_id=subjectAltNameExtDefaultImpl
policyset.serverCertSet.9.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_0=$request.requestor_email$
policyset.serverCertSet.9.default.params.subjAltExtType_0=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.SAN1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=http://www.server.example.com
policyset.serverCertSet.9.default.params.subjAltExtType_2=URIName
policyset.serverCertSet.9.default.params.subjAltExtType_3=OtherName
policyset.serverCertSet.9.default.params.subjAltExtPattern_3=
(IA5String)1.2.3.4,$server.source$
policyset.serverCertSet.9.default.params.subjAltExtSource_3=UUID4
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_3=true
policyset.serverCertSet.9.default.params.subjAltExtType_4=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_4=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_4=
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=4
```

Subject Alternative Name 扩展默认检查配置集属性的证书请求。如果请求包含属性，则配置集会读取其值并将其设置在 扩展名中。如果配置了基于 LDAP 的身份验证，则 Subject Alternative Name 扩展默认为从 LDAP 目录中插入属性值。添加到证书中的扩展包含所有配置的属性。

可以与 Subject Alternative Name 扩展默认使用的变量列在 表 B.20 “在主题备用名称中插入值的变量” 中。

表 B.20. 在主题备用名称中插入值的变量

策略设置令牌	描述
$request.auth_token.cn$	请求证书的用户的 LDAP 通用名称(cn)属性。
$request.auth_token.mail$	请求证书的用户的 LDAP 电子邮件（邮件）属性值。
$request.auth_token.tokenCertSubject$	证书主题名称。
$request.auth_token.uid$	请求证书的用户的 LDAP 用户 ID(uid)属性。
$request.auth_token.user$	
$request.auth_token.userDN$	请求证书的用户 DN。
$request.auth_token.userid$	请求证书的用户的用户 ID 属性的值。
$request.uid$	请求证书的用户的用户 ID 属性的值。
$request.profileRemoteAddr$	发出请求的用户的 IP 地址。根据客户端，这可以是 IPv4 或 IPv6 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如： 0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1 .68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:F F:FF:FF:FF:FF: FF
$request.profileRemoteHost$	用户机器的主机名或 IP 地址。主机名可以是完全限定域名和协议，如 http://server.example.com。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如： 0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1 .68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:F F:FF:FF:FF:FF: FF

策略设置令牌	描述
$request.requestor_email$	提交请求的人的电子邮件地址。
$request.requestowner$	提交请求的人员。
$request.subject$	发布证书的实体的主题名称 DN。例如，uid=jsmith，e=jsmith@example.com。
$request.tokencuid$	用于请求注册的智能卡唯一 ID(CUID)
$request.upn$	Microsoft UPN。它的格式为 (UTF8String)1.3.6.1.4.1.311.20.2.3，$request.upn$。
$server.source$	指示服务器在主题名称中生成版本 4 个 UUID（随机号）组件。这始终具有格式 (IA5String)1.2.3.4,$server.source$.

可以为单个扩展设置多个属性。subjAltNameNumGNs 参数控制在证书中添加所需列出属性的数量。此参数必须添加到自定义配置集中，可能需要在默认配置集中修改，以根据需要包含任意数量的属性。在 *例 B.1 “默认主题备用名称扩展配置”* 中，subjAltNameNumGNs 设置为 3，以插入 RFC822Name、DNSName 和 URIName 名称（通用名称 _ 0、_1 和 _2）。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 *第 B.2.4 节 “扩展约束”*。
- 无限制；请参阅 *第 B.2.8 节 “No Constraint”*。

表 B.21. 主题名称扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键；选择 false 来标记非关键。

参数	描述
pattern	指定要在扩展中包含的 request 属性值。属性值必须符合任何受支持的通用名称类型。如果服务器在请求中找到 属性，它将在扩展中设置属性值，并为证书添加扩展。如果指定了多个属性，且请求中没有包括任何属性，服务器不会为证书添加 Subject Alternative Name 扩展。permissible 值是证书请求中包含的 request 属性。例如，$request.requestor_email$.
类型	指定 request 属性的一般名称类型。 • 如果 request-attribute 值是一个采用 local-part@domain 格式的电子邮件地址，请选择 RFC822Name。例如：jdoe@example.com • 如果 request-attribute 值是一个 X.500 目录名称，则选择 DirectoryName，类似于证书中的主题名称。例如： cn=Jane Doe, ou= Sales Dept, o=Example Corporation, c=US. • 如果 request-attribute 值是一个 DNS 名称，请选择 DNSName。例如：corpDirectory.example.com。 • 如果 request-attribute 值是一个 EDI party 名称，请选择 EDIpartyName。例如，Sartor 公司示例。 • 如果 request-attribute 值是一个非关系 URI，则选择 URIName，其中包含一个方案，如 http，以及一个完全限定域名或 IP 地址。例如 ：http://hr.example.comCertificateCerti

参数	描述
	ficate Systemnbsp;System 支持 IPv4 和 IPv6 地址。 • 如果 request-attribute 值是一个以点分隔的数字组件表示法指定的有效 IP 地址，请选择 IPAddress。例如 ：128.21.39.40。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • 如果 request-attribute 值是一个唯一且有效的 OID，请选择以点分隔的数字组件表示法指定的 OID。例如，1.2.3.4.55.6.5.99。 • 选择 OtherName 作为具有任何其他格式的名称。它支持 PrintableString、IA5String、UTF8String、BMPString、any、和 KerberosName。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。 OtherName 的格式 （类型）oid，字符串。例如，(IA5String)1.2.3.4,MyExample.
源	指定用于生成 ID 的标识源或协议。唯一支持的来源是 UUID4，它会生成一个随机数字来创建 UUID。
组件数量(NumGN)	指定在主题备用名称中包含的名称组件数量。

B.1.24. 主题目录属性扩展默认值

此默认值为证书附加 Subject Directory 属性扩展。Subject Directory 属性扩展为证书主体提供任何所需的目录属性值。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制；请参阅 第 B.2.8 节 “No Constraint”。

表 B.22. 主题目录属性扩展默认配置参数

参数	描述
Critical	选择 true 来标记这个扩展关键；选择 false 来标记非关键。
名称	属性名称；这可以是任何 LDAP 目录属性，如 cn 或 mail。
pattern	指定要在扩展中包含的 request 属性值。属性值必须符合属性的允许值。如果 server 找到 属性，它将在扩展中设置属性值，并为证书添加扩展。如果指定了多个属性，并且请求中不存在任何属性，服务器不会将 Subject Directory Attributes 扩展添加到证书。例如，$request.requestor_email$.
启用	设置该属性是否能够添加到证书中。选择 true 来启用属性。

B.1.25. 主题信息扩展默认

实施注册默认策略，该策略在证书模板中填充使用者信息访问扩展。此扩展指示了如何访问该扩展出现在证书中的主题的信息和服务。

参数	描述
Critical	这个扩展应该不是关键。
subjInfoAccessNumADs	证书中包含的信息访问部分数量。

参数	描述
subjInfoAccessADMethod_n	*访问方法的 OID。*
subjInfoAccessADMethod_n	*访问方式类型。* • **URIName** • *目录名称* • **DNS 名称** • *第三方名称* • **IP 地址** • **OID 名称** • **RFC822Name**
subjInfoAccessADLocation_n	*基于类型 **subjInfoAccessADMethod_n** 的位置* ***i.e.，URI 名称的 URL。**

参数	描述
subjInfoAccessADEnable_n	选择 true 以启用这个扩展；选择 false 来禁用这个扩展。

B.1.26. 主题键标识符默认

此默认向证书附加一个 Subject Key Identifier 扩展。扩展标识包含特定公钥的证书，该密钥从具有相同主题名称的几项中标识证书。

有关这个扩展的常规信息，请参阅 第 B.3.16 节 “subjectKeyIdentifier”。

如果启用，配置集在扩展尚不存在时为注册请求添加 Subject Key Identifier 扩展。如果请求中存在扩展（如 CRMF 请求），则默认替换扩展。代理批准手动注册请求后，配置集接受已经存在的任何 Subject Key Identifier Extension。

这个默认没有参数。如果使用，这个扩展会包含在证书中，并包含公钥信息。

以下限制可使用此默认值定义：

- 扩展约束；请参阅 第 B.2.4 节 “扩展约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.27. 主题名称默认

这个默认会将服务器端可配置主体名称附加到证书请求。静态主题名称用作证书中的主题名称。

以下限制可使用此默认值定义：

- 主题名称约束；请参阅 第 B.2.11 节 “主题名称约束”。
- 唯一的 Subject Name Constraint; 请参阅 第 B.2.13 节 “唯一的 Subject Name

Constraint”。

- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.23. 主题名称默认配置参数

参数	描述
名称	指定此证书的主题名称。

如果您需要从 UidPwdDirAuth 插件获取使用 DNPATTERN 值的证书主题名称，那么请将配置集配置为使用 Subject Name Default 插件，并将 Name 参数替换为 AuthToken 中的 "Subject Name"，如下所示。

```
policyset.userCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.userCertSet.1.default.name=Subject Name Default
policyset.userCertSet.1.default.params.name=$request.auth_token.tokenCertSubject$
```

B.1.28. 用户密钥默认值

此默认将用户提供的密钥附加到证书请求中。这是必需默认值。密钥是注册请求的一部分。

以下限制可使用此默认值定义：

- 键约束; 请参阅 第 B.2.5 节 “键约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.29. 用户签名算法

此默认实施注册默认配置集，该配置集填充证书请求的用户提供的签名算法。如果证书配置集中包含，允许用户根据约束集选择证书的签名算法。

不会在注册表单中添加签名算法选项，但可以提交包含此信息的请求。

以下限制可使用此默认值定义：

- 签名算法; 请参阅 第 B.2.10 节 “签名算法”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.30. 用户主题名称默认

此默认将用户提供的主题名称附加到证书请求。如果包含在证书配置集中，它允许用户提供证书的主体名称，这取决于限制集。这个扩展会保留在签发证书时在原始证书请求中指定的主题名称。

以下限制可使用此默认值定义：

- 主题名称约束 ; 请参阅 第 B.2.11 节 “主题名称约束”。
- 唯一的 Subject Name Constraint; 请参阅 第 B.2.13 节 “唯一的 Subject Name Constraint”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.31. 用户有效期默认值

此默认向用户提供证书请求的有效性。如果包含在证书配置集中，它允许用户提供有效期周期，这取决于限制集。这个默认配置集保留在签发证书时，在原始证书请求中用户定义的有效期期。

不提供向注册表单添加用户提供的有效日期的输入，但可以提交包含此信息的请求。

以下限制可使用此默认值定义：

- 有效期约束 ; 请参阅 第 B.2.14 节 “有效期约束”。

- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

B.1.32. 用户补充默认设置

用户补充的扩展默认类会在证书请求中使用用户定义的任何证书扩展填充证书。这要求用户提交满足特定标准或提供某些信息的证书请求，因为配置集在注册证书前需要特定的扩展。

WARNING

请特别小心设置此扩展默认设置，因为它允许用户在证书请求中指定扩展。如果使用了这个默认值，则 Red Hatnbsp；强烈建议您使用与扩展对应的约束来尽可能减少用户补充的 Extension 默认可能。

用户定义的扩展会根据设置的任何约束进行验证，因此可以限制扩展类型（通过 Extension Constraint）或为密钥和其他基本约束设置规则，如是否是一个 CA 证书。

注意

如果在带有对应 OID(Extension Constraint)的配置集上设置此扩展，那么通过该配置集处理的任何证书请求 都必须 执行指定的扩展，或者请求被拒绝。

如果在勘误 RHSA 2008:0500 之前使用 User Supplied Extension Default 启用了证书配置集，则必须编辑此配置集以支持证书请求中的用户提供的扩展。应用 userExtensionDefaultImpl 默认，如示例所示。给定的 OID 用于基本限制扩展约束。

```
policyset.set1.p6.default.class_id=userExtensionDefaultImpl
policyset.set1.p6.default.name=User Supplied Extension Default
policyset.set1.p6.default.params.userExtOID=2.5.29.19
```

CA 通过以下三种方法之一使用用户所提供的扩展扩展默认值处理注册：

- 如果在证书请求和默认值中指定扩展的 OID，则扩展由约束验证并应用到证书。
-

如果在请求中给出了扩展的 OID，但没有在配置集中的"用户补充扩展"中指定，则用户指定的扩展名将被忽略，且证书在没有该扩展的情况下被成功注册。

- 如果在带有对应 OID(Extension Constraint)的配置集上设置此扩展，那么通过该配置集处理的任何证书请求 都必须 执行指定的扩展，或者请求被拒绝。

包含用户定义的扩展的证书请求必须提交到配置集。但是，证书注册表单用户没有任何输入字段来添加用户提供的扩展。在不提供扩展的情况下提交证书请求会失败。

例 B.2 “用户为扩展密钥使用扩展扩展默认值” 使用扩展密钥使用约束将用户补充扩展默认值添加到配置集中。userExtOID 参数中指定的 OID 用于扩展密钥使用范围。

例 B.2. 用户为扩展密钥使用扩展扩展默认值

```
policyset.set1.2.constraint.class_id=extendedKeyUsageExtConstraintImpl
policyset.set1.2.constraint.name=Extended Key Usage Extension
policyset.set1.2.constraint.params.exKeyUsageCritical=false
policyset.set1.2.constraint.params.exKeyUsageOIDs=1.3.6.1.5.5.7.3.2,1.3.6.1.5.5.7.3.4
policyset.set1.2.default.class_id=userExtensionDefaultImpl
policyset.set1.2.default.name=User Supplied Extension Default
policyset.set1.2.default.params.userExtOID=2.5.29.37
```

在 例 B.2 “用户为扩展密钥使用扩展扩展默认值” 中，尽管用户补充一个 Extension Default 允许用户指定扩展密钥使用 Extension(2.5.29.37)，但约束会将用户请求限制为只有 SSL 客户端身份验证 (1.3.6.1.5.5.7.3.2)和电子邮件保护(1.3.6.5.5.7.3.4)使用。

编辑配置集请参考 第 3.2 节 “设置证书配置集”。

例 B.3. CSR 中的多个用户提供的扩展

RHCS 注册配置集框架允许在同一配置集中定义多个用户补充扩展。例如，可以指定以下组合：

- 对于扩展密钥使用：

```
policyset.serverCertSet.2.constraint.class_id=extendedKeyUsageExtConstraintImpl
policyset.serverCertSet.2.constraint.name=Extended Key Usage Extension
policyset.serverCertSet.2.constraint.params.exKeyUsageCritical=false
policyset.serverCertSet.2.constraint.params.exKeyUsageOIDs=1.3.6.1.5.5.7.3.2,1.3.6.1.5.5.7.3.4
```

```
policyset.serverCertSet.2.default.class_id=userExtensionDefaultImpl
policyset.serverCertSet.2.default.name=User Supplied Extension Default
policyset.serverCertSet.2.default.params.userExtOID=2.5.29.37
```

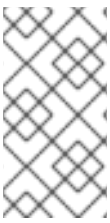
- ***对于主要使用范围：***

  使用以下格式，您可以应用扩展参数的策略：

  - ***CSR: value = "true"中必须已存在***
  - ***CSR: value = "false"中 不能 存在***
  - ***是可选的:value = "-"***

  例如：

```
policyset.serverCertSet.13.constraint.class_id=keyUsageExtConstraintImpl
policyset.serverCertSet.13.constraint.name=Key Usage Extension Constraint
policyset.serverCertSet.13.constraint.params.keyUsageCritical=-
policyset.serverCertSet.13.constraint.params.keyUsageCrlSign=false
policyset.serverCertSet.13.constraint.params.keyUsageDataEncipherment=-
policyset.serverCertSet.13.constraint.params.keyUsageDecipherOnly=-
policyset.serverCertSet.13.constraint.params.keyUsageDigitalSignature=-
policyset.serverCertSet.13.constraint.params.keyUsageEncipherOnly=-
policyset.serverCertSet.13.constraint.params.keyUsageKeyAgreement=true
policyset.serverCertSet.13.constraint.params.keyUsageKeyCertSign=-
policyset.serverCertSet.13.constraint.params.keyUsageKeyEncipherment=-
policyset.serverCertSet.13.constraint.params.keyUsageNonRepudiation=-
policyset.serverCertSet.13.default.class_id=userExtensionDefaultImpl
policyset.serverCertSet.13.default.name=User Supplied Key Usage Extension
policyset.serverCertSet.13.default.params.userExtOID=2.5.29.15
```



注意

有关如何使用用户定义的扩展属性创建 CSR 的示例，请参阅 第 5.2.1.1.2 节 “使用 certutil 使用用户定义的扩展来创建 CSR”。

B.1.33. 有效性默认值

这个默认会将服务器端可配置的有效期期附加到证书请求中。

以下限制可使用此默认值定义：

- 有效期约束；请参阅 第 B.2.14 节 “有效期约束”。
- 无限制; 请参阅 第 B.2.8 节 “No Constraint”。

表 B.24. 有效的默认配置参数

参数	描述
range	指定此证书的有效性周期。
startTime	根据当前时间，设置有效期时。

B.2. 约束参考

约束用于定义证书的允许内容以及与该内容关联的值。本节列出了预定义的约束，以及每个部分的完整定义。

B.2.1. 基本限制扩展约束

Basic Constraints 约束会检查证书请求中是否满足此约束中设置的条件。

表 B.25. 基本限制约束配置参数

参数	描述
basicConstraintsCritical	指定扩展是否可以标记为关键或非关键。选择 true 来标记此扩展关键; 选择 false 以阻止此扩展标记为关键。选择一个连字符 -，表示没有关键性首选项。
basicConstraintsIsCA	指定证书主题是否为 CA。选择 true 需要这个参数的值为 true （是 CA）；选择 false 以禁止此参数使用 true ；选择一个连字符 - - 以指示没有为此参数放置限制。

参数	描述
basicConstraintsMinPathLen	指定允许的路径长度的最低允许，下可链的 CA 证书的最大数量（提交至）发出的从属 CA 证书。路径长度会影响证书验证过程中使用的 CA 证书数量。链从验证和移动的最终证书开始。 如果以长期证书设定扩展，则此参数无效。 允许的可能值为 0 或 n。该值必须小于 CA 签名证书的 Basic Constraints 扩展中指定的路径长度。 0 指定，在签发的从属 CA 证书下不允许从属 CA 证书；在路径中只能遵循最终用户证书。 n 必须是大于零的整数。这是最低限额的 CA 证书数量，允许位于所用的下级 CA 证书。

参数	描述
basicConstraintsMaxPathLen	指定最大允许的路径长度，在下面链接的最大 CA 证书数（提交至）发出的从属 CA 证书数。路径长度会影响证书验证过程中使用的 CA 证书数量。链从验证和移动的最终证书开始。 如果以长期证书设定扩展，则此参数无效。 允许的可能值为 0 或 n。该值必须大于 CA 签名证书的 Basic Constraints 扩展中指定的路径长度。 0 指定，在签发的从属 CA 证书下不允许从属 CA 证书；在路径中只能遵循最终用户证书。 n 必须是大于零的整数。这是在使用从属 CA 证书下允许的最大下级 CA 证书数。 如果字段为空，则路径长度默认为由签发者证书中 Basic Constraints 扩展中设置的路径长度决定的值。如果签发者的路径长度不受限制，则从属 CA 证书中的路径长度也会无限。如果签发者的路径长度是一个大于零的整数，则从属 CA 证书中的路径长度被设置为比签发者的路径长度小的值之一；例如，如果签发者的路径长度为 4，则子协调 CA 证书中的路径长度被设置为 3。

参数	描述

B.2.2. CA Validity Constraint

CA 有效期约束会检查证书模板中的有效周期是否在 CA 的有效性期内。如果证书的有效性期超出 CA 证书的有效性期，则约束将被拒绝。

B.2.3. 扩展密钥使用扩展约束

Extended Key Usage 扩展约束会检查证书是否满足此约束中设置的条件。

表 B.26. 扩展密钥使用扩展约束配置参数

参数	描述
exKeyUsageCritical	当设置为 true 时，扩展名可以标记为 critical。当设置为 false 时，扩展可以标记为非关键。
exKeyUsageOIDs	指定用于标识密钥使用目的的允许 OID。可以在以逗号分隔的列表中添加多个 OID。

B.2.4. 扩展约束

此约束实施常规扩展约束。它将检查是否存在扩展。

表 B.27. 扩展约束

参数	描述
extCritical	指定扩展是否可以标记为关键或非关键。选择 true 来标记扩展名严重；选择 false 来标记它非关键。选择 - 不强制选择。
extOID	证书必须存在的扩展名的 OID 才能通过约束。

B.2.5. 键约束

这个约束会检查 RSA 密钥的密钥大小，以及 EC 密钥的 elliptic curve 的名称。与 RSA 密钥一起使用时，key Parameters 参数包含以逗号分隔的法律密钥大小列表，而 EC 键则包括了用逗号分开的可用 ECC curves 列表。

表 B.28. 键约束配置参数

参数	描述
keyType	指定密钥类型，默认为，并使用 RSA 密钥系统。选择是 rsa 和 ec。如果指定了密钥类型且未由系统标识，则约束将被拒绝。
KeyParameters	定义特定密钥参数。为键设置的参数因 键 类型而异。 • 使用 RSA 密钥，key Parameters 参数包含以逗号分隔的法律密钥大小列表。 • 使用 ECC 键时，keyParameters 参数包含以逗号分隔的可用 ECC curves 列表。

B.2.6. 主要使用扩展约束

Key Usage 扩展约束会检查证书请求中的密钥用量约束是否满足此约束中设置的条件。

表 B.29. 主要使用扩展约束配置参数

参数	描述
keyUsageCritical	选择 true 来标记此扩展关键；选择 false 来标记它非关键。选择 - 无首选项。
keyUsageDigitalSignature	指定是否为 SSL 客户端证书和 S/MIME 签名证书签名。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。

参数	描述
kleyUsageNonRepudiation	指定是否设置 S/MIME 签名证书。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。 **WARNING** 使用这个位是简介的。在为任何证书设置前，请仔细考虑其使用的合法后果。
keyEncipherment	指定是否为 SSL 服务器证书和 S/MIME 加密证书设置扩展。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。
keyUsageDataEncipherment	指定在使用主体的公钥加密用户数据而非密钥材料时，是否设置扩展。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。
keyUsageKeyAgreement	指定每当将主题的公钥用于密钥协议时，是否设置扩展。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。
keyUsageCertsign	指定扩展是否适用于所有 CA 签名证书。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。
keyUsageCRLSign	指定是否为用于签署 CRL 的 CA 签名证书的扩展。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。

参数	描述
keyUsageEncipherOnly	指定在仅用于加密数据的公钥时是否设定扩展。如果设置了这个位，则还应设置 keyUsageKeyAgreement。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。
keyUsageDecipherOnly	指定是否只用于压缩数据，是否设置扩展。如果设置了这个位，则还应设置 keyUsageKeyAgreement。选择 true 以将其标记为 set；选择 false 以保留此设置；选择一个连字符 - 以指示没有为此参数放置限制。

B.2.7. Netscape Certificate Type Extension Constraint

WARNING

这个约束已过时。使用 Key Usage 扩展或扩展密钥使用扩展扩展，而不是使用 Netscape 证书类型扩展。

Netscape 证书类型扩展约束会检查证书请求中的 Netscape 证书类型扩展是否满足此约束中设置的条件。

B.2.8. No Constraint

这个约束没有实现约束。当选择默认值时，默认不会有限制。

B.2.9. renewal Grace Period Constraint

当用户可以根据其过期日期续订证书时，renew Period Constraint 设置规则。例如，在证书过期前或过期时间过后，用户无法续订证书。

使用此约束时要记住的一个重要事项是，在 原始注册配置文件 上设置这个约束，而不是续订配置文件。续订宽限期的规则是原始证书的一部分，并适用于后续的续订。

这个约束只适用于 No Default 扩展。

表 B.30. renewal Period Constraint 配置参数

参数	描述
renewal.graceAfter	在证书过期 后 设置该周期（以天为单位），以供续订。如果证书已过期了该时间，则拒绝续订请求。如果没有给定值，则没有限制。
renewal.graceBefore	在证书过期 之前设置该 周期，以便进行续订。如果证书没有接近其过期日期，则拒绝续订请求。如果没有给定值，则没有限制。

B.2.10. 签名算法

Signing Algorithm 约束会检查证书请求中的签名算法是否满足此约束中设置的条件。

表 B.31. 签名算法 Constraint 配置参数

参数	描述

参数	描述
signingAlgsAllowed	设置可指定为为证书签名的签名算法。这些算法可以是以下任意一种或全部算法： • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA • SHA256withEC • SHA384withEC • SHA512withEC

B.2.11. 主题名称约束

Subject Name 约束会检查证书请求中的主题名称是否满足条件。

表 B.32. subject Name Constraint 配置参数

参数	描述
pattern	指定用于构建主题 DN 的正则表达式或其他字符串。

主题名称和正则表达式

Subject Name Constraint 的正则表达式与匹配正则表达式的 Java 工具匹配。这些正则表达式的格式列在 中 https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html。这允许通配符（如星号(1)）搜索任意字符和句点(.)搜索任何类型字符。

例如，如果主题名称约束的模式设置为 uid=.*，则证书配置集框架会检查证书请求中的主题名称是否与模式匹配。uid=user、o=Example、c=US 满足模式 uid=.* 等主题名称。subject name cn=user, o=example,c=US 不满足该模式。UID=.* 表示主题名称必须以 uid 属性开头； period-asterisk(.*)通配符允许任何 type 和 uid 后面的字符数。

可能需要内部模式，如 .*ou=Engineering.*，它要求带有任意字符串的 ou=Engineering 属性。这会显示 cn=jdoe,ou=internal,ou=west coast,ou=engineering,o="Example Corp",st=NC 以及 uid=bjensen,ou=engineering,dc=example,dc=com。

最后，也可以通过在选项间设置管道符号(|)来允许作为字符串或另一个字符串的请求。例如，要允许包含 ou=engineering ,ou=engineering, ou= engineering,o="Example Corp" 的主题名称，其模式是 .*ou=engineering,ou= people.* | .*ou=engineering,o="Example Corp".*.

注意

对于构造使用特殊字符的模式，如句点（），请使用反斜杠\转义字符。例如，要搜索字符串 o="Example Inc."，请将特征设置为 o="Example Inc\. "。

证书请求中的 subject Name 和 UID 或 CN

用于构建主题 DN 的模式也可以基于请求证书的用户的 CN 或 UID。Subject Name Constraint 设置 CN（或 UID）的 patter，以在证书请求的 DN 中识别，然后在该 CN 上构建 Subject Name Default 构建，以使用预定义的目录树创建证书的主题 DN。

例如，使用证书请求的 CN：

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=CN=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=CN=$request.req_subject_name.cn$,DC=example,
DC=com
```

B.2.12. 唯一的键约束

此约束会检查公钥是否是唯一的。

表 B.33. 唯一的键限制参数

参数	描述
allowSameKeyRenewal	请求被视为续订，如果此参数设置为 true，如果公钥不唯一，则接受请求，如果主题 DN 与现有证书匹配，则接受请求。但是，如果公钥是重复的，并且与现有对象 DN 不匹配，则请求将被拒绝。 当 参数设置为 false 时，重复的公钥请求将被拒绝。

B.2.13. 唯一的 Subject Name Constraint

unique Subject Name 约束会限制服务器以相同的主题名称发布多个证书。提交证书请求时，服务器会自动检查针对其他颁发的证书 nicknames 检查 nickname。此约束可应用于通过结束日期的页面进行证书注册和续订。

除非证书已过期或撤销，否则证书不能具有相同的 subject 名称。因此，活跃的证书无法共享主题名称，但有一个例外：如果证书有不同的密钥使用位，则它们可以共享相同的主题名称，因为它们有不同的使用。

表 B.34. 唯一的 Subject Name Constraint 配置参数

参数	描述
enableKeyUsageExtensionChecking	允许证书拥有相同主题名称的可选设置，只要其密钥使用设置不同。这是 true 或 false。默认为 true，它允许重复的主题名称。

B.2.14. 有效期约束

Validity 约束会检查证书请求中的有效期周期是否满足相关的标准。

提供的参数必须是健全的值。例如，一个 notBefore 参数提供了一个未接受传递的时间，以及一个未接受 notBefore 的时间超过 notBefore 的 notAfter 参数。

表 B.35. 有效约束配置参数

参数	描述
range	有效周期的范围。这是一个整数，用于设置天数。notBefore 和 notAfter 时间之间的差别（在天数内）必须小于范围值，否则此约束将被拒绝。
notBeforeCheck	验证范围是否在宽限期内。当 NotBeforeCheck 布尔值参数设为 true 时，系统将检查 notBefore 时间不大于当前时间加上 notBeforeGracePeriod 值。如果 notBeforeTime 不在当前时间和 notBeforeGracePeriod 值之间，则这个约束将被拒绝。
notBeforeGracePeriod	宽限期（以秒为单位）。如果 notBeforeTime 不在当前时间和 notBeforeGracePeriod 值之间，则这个约束将被拒绝。只有在 notBeforeCheck 参数设置为 true 时才会检查此约束。
notAfterCheck	给定时间是否在到期期后。当 notAfterCheck 布尔值参数设为 true 时，系统会检查 notAfter 时间大于当前时间。如果当前时间超过 notAfter 时间，则此约束将被拒绝。

B.3. 标准 X.509 V3 证书扩展参考

X.509 v3 证书包含一个扩展字段，允许向证书中添加任意数量的附加字段。证书扩展提供了向证书添加信息（如替代主题名称和使用量限制）的方法。旧版 Netscape 服务器，如 Red Hat Directory Server and Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System，在 PKIX 部分定义 1 标准之前已定义了特定于 Netscape 的扩展。

以下是包含 X.509 v3 扩展的证书的 部分示例。CertificateCertificate Systemnbsp;System 可以以可读用户打印格式显示证书，如下所示。如本例中，证书扩展出现在序列中，且每个证书只能有一个特定扩展实例；例如，证书只能包含一个主题密钥标识符扩展。支持这些扩展的证书有版本 0x2 （对应于版本 3）。

例 B.4. Pretty-Print Certificate Extensions 示例

```
Data:
  Version:  v3
  Serial Number: 0x1
  Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
  Issuer: CN=Certificate Manager,OU=netscape,O=ExampleCorp,L=MV,ST=CA,C=US
  Validity:
    Not Before: Friday, February 21, 2005 12:00:00 AM PST America/Los_Angeles
    Not  After: Monday, February 21, 2007 12:00:00 AM PST America/Los_Angeles
  Subject: CN=Certificate Manager,OU=netscape,O=ExampleCorp,L=MV,ST=CA,C=US
  Subject Public Key Info:
    Algorithm: RSA - 1.2.840.113549.1.1.1
    Public Key:
      Exponent: 65537
      Public Key Modulus: (2048 bits) :
        E4:71:2A:CE:E4:24:DC:C4:AB:DF:A3:2E:80:42:0B:D9:
        CF:90:BE:88:4A:5C:C5:B3:73:BF:49:4D:77:31:8A:88:
        15:A7:56:5F:E4:93:68:83:00:BB:4F:C0:47:03:67:F1:
        30:79:43:08:1C:28:A8:97:70:40:CA:64:FA:9E:42:DF:
        35:3D:0E:75:C6:B9:F2:47:0B:D5:CE:24:DD:0A:F7:84:
        4E:FA:16:29:3B:91:D3:EE:24:E9:AF:F6:A1:49:E1:96:
        70:DE:6F:B2:BE:3A:07:1A:0B:FD:FE:2F:75:FD:F9:FC:
        63:69:36:B6:5B:09:C6:84:92:17:9C:3E:64:C3:C4:C9
  Extensions:
    Identifier: Netscape Certificate Type - 2.16.840.1.113730.1.1
      Critical: no
      Certificate Usage:
        SSL CA
        Secure Email CA
        ObjectSigning CA
    Identifier: Basic Constraints - 2.5.29.19
      Critical: yes
      Is CA: yes
      Path Length Constraint: UNLIMITED
    Identifier: Subject Key Identifier - 2.5.29.14
      Critical: no
      Key Identifier:
        3B:46:83:85:27:BC:F5:9D:8E:63:E3:BE:79:EF:AF:79:
        9C:37:85:84
    Identifier: Authority Key Identifier - 2.5.29.35
      Critical: no
      Key Identifier:
        3B:46:83:85:27:BC:F5:9D:8E:63:E3:BE:79:EF:AF:79:
        9C:37:85:84
    Identifier: Key Usage: - 2.5.29.15
      Critical: yes
      Key Usage:
        Digital Signature
        Key CertSign
        Crl Sign
  Signature:
    Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
    Signature:
      AA:96:65:3D:10:FA:C7:0B:74:38:2D:93:54:32:C0:5B:
      2F:18:93:E9:7C:32:E6:A4:4F:4E:38:93:61:83:3A:6A:
```

A2:11:91:C2:D2:A3:48:07:6C:07:54:A8:B8:42:0E:B4:
E4:AE:42:B4:B5:36:24:46:4F:83:61:64:13:69:03:DF:
41:88:0B:CB:39:57:8C:6B:9F:52:7E:26:F9:24:5E:E7:
BC:FB:FD:93:13:AF:24:3A:8F:DB:E3:DC:C9:F9:1F:67:
A8:BD:0B:95:84:9D:EB:FC:02:95:A0:49:2C:05:D4:B0:
35:EA:A6:80:30:20:FF:B1:85:C8:4B:74:D9:DC:BB:50

对象标识符 (OID)是识别唯一对象（如证书扩展名或公司证书实践声明）的数字字符串。CertificateCertificate Systemnbsp;System 附带一组特定于扩展的配置集插件模块，该模块使 X.509 证书扩展添加到服务器问题中。有些扩展包含用来指定 OID 的字段。

PKIX 标准建议所有对象（如扩展和语句）以 OID 的形式包含在证书中。这促进了共享网络上的机构之间的互操作性。如果将发布在共享网络上使用的证书，请将 OID 前缀注册到适当的注册机构。

OID 由国际标准机构(ISO)注册机构控制。在某些情况下，此授权由 ISO 委派给区域注册机构。美国美国国家标准研究所(ANSI)管理此注册。

使用注册到另一个机构或无法注册 OID 的 OID 可能会发生法律后果，具体取决于情况。注册可能取决于费用。如需更多信息，请联系相应的注册机构。

要为自定义对象定义或分配 OID，请知道公司的 arc，这是私有企业的 OID。如果公司没有rc，则需要获得一项。http://www.alvestrand.no/objectid/ 包含有关注册和使用 OID 的更多信息。

例如，名为 Netscape Certificate Comment 的扩展 Netscape-defined OID 为 2.16.840.1.113730.1.13。为这个扩展分配的 OID 是分级的，包括前一个 Netscape company arc 2.16.840.1。OID 定义条目是 http://www.alvestrand.no/objectid/2.16.840.1.113730.1.13.html。

如果证书中存在 OID 扩展并标记为关键，则验证证书的应用程序必须能够解释该扩展，包括任何可选的限定符，或者它必须拒绝该证书。由于所有应用程序都不太可能解释以 OID 格式嵌入的公司的自定义扩展，因此 PKIX 标准建议扩展始终标记为非关键。

本节总结了定义为互联网 X.509 版本 3 标准的一部分定义的扩展类型，并指示 PKIX 工作组推荐的类型。

本参考总结了有关每个证书的重要信息。有关完整详情，请参阅来自 ITU 的 X.509 v3 标准，以及 互联网 X.509 公共密钥基础架构 - 证书和密钥配置文件(RFC 3 280)，请访问 RFC 3280。扩展描述引用标准草案的 RFC 和部分号，还会为每个扩展提供对象标识符(OID)。

证书中的每个扩展都可以指定为关键或非关键。使用证书系统（如 Web 浏览器）时，如果证书遇到无法识别的关键扩展，则必须拒绝证书。但是，如果无法识别，将无法忽略非关键扩展名。

B.3.1. authorityInfoAccess

授权信息访问扩展指示了如何和在哪里访问证书签发者的信息。该扩展包含一个 accessMethod 和 accessLocation 字段。accessMethod 由 OID 指定，在 accessLocation 中有关 named 签发者的信息的类型和格式。

PKIX 第 1 部分定义一个 accessMethod (id-ad-caIssuers)的列表，以获取 CA 链中签发的证书列表，超过使用扩展名的证书。accessLocation 字段随后通常包含一个 URL，表示用于检索列表的位置和协议（LDAP、HTTP 或者 FTP）。

Online Certificate Status Protocol(RFC 2560)，在 RFC 2560 上提供，定义一个 accessMethod (id-ad-ocsp)来使用 OCSP 来验证证书。accessLocation 字段随后包含一个 URL，表示用于访问能够验证证书的 OCSP 响应者的位置和协议。

OID

1.3.6.1.5.5.7.1.1

严重程度

这个扩展必须是非关键。

B.3.2. authorityKeyIdentifier

Authority Key Identifier 扩展标识与用于为证书签名的私钥对应的公钥。当签发者有多个签名密钥时，这个扩展很有用，比如当 CA 证书被续订时。

扩展由以下之一或两个组成：

- 一个显式键标识符，在 keyIdentifier 字段中设置
- 在 authorityCertIssuer 字段和序列号中设置的签发者，在 authorityCertSerialNumber 字段中设置，标识证书

如果 keyIdentifier 字段存在，它将用来选择具有匹配 subjectKeyIdentifier 扩展名的证书。如果 颁发机构CertIssuer 和 authorityCertSerialNumber 字段存在，则会使用它们通过 签发者 和 serialNumber 来识别正确的证书。

如果这个扩展不存在，则单独使用签发者名称来识别签发者证书。

PKIX 第 1 部分要求除自签名 root CA 证书外的所有证书都必须此扩展。如果尚未建立密钥标识符，PKIX 建议需要指定 授权CertIssuer 和 authorityCertSerialNumber 字段。这些字段允许构建完整的证书链，方法是与主题证书中的 SubjectName 和 Certificate Ser 和 CertificateSers suer 和 authorityCertSer 标识符扩展中的授权认证链匹配。

OID

2.5.29.35

严重程度

这个扩展总是非关键，且总是被评估。

B.3.3. basicConstraints

这个扩展用于在证书链验证过程中用来识别 CA 证书并应用证书链路径长度限制。对于所有 CA 证书，cA 组件应设置为 true。PKIX 建议此扩展不应出现在最终用户证书中。

如果存在 pathLenConstraint 组件，则其值必须大于到目前为止已处理的 CA 证书数量，从最终用户证书开始并移动链。如果省略 pathLenConstraint，则链中的所有更高级别的 CA 证书在扩展存在时不得包含这个组件。

OID

2.5.29.19

严重程度

PKIX 第 1 部分要求该扩展标记为关键。无论其关键性如何，都将评估此扩展。

B.3.4. certificatePoliciesExt

证书策略扩展定义了一个或多个策略，各自由 OID 和可选的限定符组成。扩展可以包括 URI 到签发者

的证书声明，或者可以嵌入签发者信息，如用户以文本形式通知。该信息可以被启用证书的应用程序使用。

如果存在此扩展，则 PKIX 第 1 部分建议仅通过 OID 标识策略，或者仅在需要时识别某些推荐的限定符。

OID

2.5.29.32

严重程度

这个扩展可能是关键或非关键。

B.3.5. CRLDistributionPoints

此扩展定义如何获取 CRL 信息。如果系统被配置为使用 CRL 发出点，则应使用它。

如果扩展包含设为 URI 的 DistributionPointName，则 URI 将假定为指向当前 CRL 的指针，出于指定的撤销原因，并将由指定 cRLIssuer 发出。URI 的预期值是为 Subject Alternative Name 扩展定义的。如果 distributionPoint 省略了原因，则 CRL 必须包含撤销所有原因。如果 发布说明 省略 cRLIssuer，则 CRL 必须由发布证书的 CA 发布。

PKIX 建议 CA 和应用程序支持此扩展。

OID

2.5.29.31

严重程度

PKIX 建议将这个扩展标记为非关键，且所有证书都支持它。

B.3.6. extKeyUsage

Extended Key Usage 扩展指示可以使用认证公钥的用途。这些用途可能还适用于 或代替 Key Usage 扩展中指示的基本目的。

Extended Key Usage 扩展必须包含 OCSP 签名 在 OCSP 响应者证书中，除非对响应者验证的证书的 CA 签名密钥也是 OCSP 签名密钥。OCSP 响应者的证书必须由 CA 直接签发响应程序将验证的证书。

密钥使用、扩展密钥用法和 Basic Constraints 扩展一起定义要使用的证书的用途。应用程序可以使用这些扩展来禁止在不当上下文中使用证书。

表 B.36 “PKIX 扩展密钥使用扩展使用” 列出 PKIX 为这个扩展定义的用途，表 B.37 “私有密钥使用扩展使用” 列表使用由 Netscape 定义。

OID

2.5.29.37

严重程度

如果此扩展标记为关键，则必须将证书用于指定的用途之一。如果它没有标记为关键，它将被当作公告字段来识别密钥，但不会限制证书对指定用途的使用。

表 B.36. PKIX 扩展密钥使用扩展使用

使用	OID
服务器身份验证	1.3.6.1.5.5.7.3.1
客户端身份验证	1.3.6.1.5.5.7.3.2
代码签名	1.3.6.1.5.5.7.3.3
电子邮件	1.3.6.1.5.5.7.3.4
时间戳	1.3.6.1.5.5.7.3.8
OCSP 签名	1.3.6.1.5.5.7.3.9[a]

使用	OID
[a] OCSP Signing 在 PKIX 第 1 部分定义，但在 RFC 2560 中，X.509 互联网公共密钥基础架构在线证书状态协议 - OCSP。	

表 B.37. 私有密钥使用扩展使用

使用	OID
证书信任列表签名	1.3.6.1.4.1.311.10.3.1
Microsoft Server Gated Crypto(SGC)	1.3.6.1.4.1.311.10.3.3
Microsoft 加密的文件系统	1.3.6.1.4.1.311.10.3.4
Netscape SGC	2.16.840.1.113730.4.1

B.3.7. issuerAltName 扩展

Issuer Alternative Name 扩展用于将互联网风格的身份与证书签发者关联。名称必须使用为 Subject Alternative Name 扩展定义的表单。

OID

2.5.29.18

严重程度

PKIX 第 1 部分建议此扩展标记为非关键。

B.3.8. keyUsage

Key Usage 扩展定义证书中包含的密钥的用途。密钥使用、扩展密钥用法和 Basic Constraints 扩展一起指定可以使用证书的用途。

如果这个扩展包含在所有中，请按如下所示设置位：

- 用于 SSL 客户端证书、S/MIME 签名证书和对象签名证书的 数字签名 (0)。
- 用于一些 S/MIME 签名证书和对象签名证书的 非 报(1)。

WARNING

此位的使用是简介的。在为任何证书设置前，请仔细考虑其使用的合法后果。

- 用于 SSL 服务器证书和 S/MIME 加密证书 的密钥 增加(2)。
- 当主体的公钥加密用户数据而非密钥材料时，数据增加(3)
- 当主题的公钥用于 密钥协议 时，密钥总会(4)。
- 所有 CA 签名证书的 keyCertSign (5)
- 用于签署 CRL 的 CA 签名证书的 cRLSign (6)。
- 如果公钥仅用于一致性数据，则强制(7)。如果设置了这个位，则还应设置 keyAgreement。
- 如果公钥仅用于减弱数据，则 decipherOnly (8)。如果设置了这个位，则还应设置 keyAgreement。

表 B.38 “证书使用和更正关键使用量” 总结典型证书使用指南。

如果存在 keyUsage 扩展并标记为关键，则会使用它强制使用证书和密钥。扩展用于限制密钥的使用；如果扩展名不存在或不存在关键，则允许所有类型的使用。

如果存在 keyUsage 扩展，则可用于从给定操作的多个证书中选择。例如，它用于为具有独立证书和密钥操作的用户区分单独的签名和加密证书。

OID

2.5.29.15

严重程度

这个扩展可能是关键或非关键。PKIX 第 1 部分建议在使用时，应将其标记为关键。

表 B.38. 证书使用和更正关键使用量

证书的目的	所需密钥使用 Bit
CA Signing	• keyCertSign • cRLSign
SSL 客户端	digitalSignature
SSL 服务器	keyEncipherment
S/MIME 签名	digitalSignature
S/MIME 加密	keyEncipherment
证书签名	keyCertSign
对象签名	digitalSignature

B.3.9. nameConstraints

此扩展只能用于 CA 证书，定义一个命名空间，其中后续证书中的所有主题名称都必须位于证书中。

OID

2.5.29.30

严重程度

PKIX 第 1 部分要求该扩展标记为关键。

B.3.10. OCSPNocheck

扩展旨在包含在 OCSP 签名证书中。扩展告知一个 OCSP 客户端，在不查询 OCSP 响应者的情况下可以信任签名证书（因为回复会再次由 OCSP 响应程序签名，客户端会再次请求签名证书的有效性状态）。这个扩展是 null-valued，它的含义由其存在或没有决定。

由于证书中存在此扩展会导致 OCSP 客户端信任使用该证书签名的响应，所以应该仔细管理使用此扩展。如果 OCSP 签名密钥被破坏，则 PKI 中验证证书的整个过程会在证书的有效性期间内被破坏。因此，使用 OCSPNocheck 的证书应该有较短的生命周期并经常续订。

OID

1.3.6.1.5.5.7.48.4

严重程度

这个扩展应该非关键。

B.3.11. policyConstraints

此扩展仅用于 CA 证书，以两种方式限制路径验证。它可用于禁止策略映射，或者要求路径中的每个证书都包含可接受的策略标识符。

PKIX 要求（若存在），此扩展永远不会包含 null 序列。必须至少存在两个可用字段之一。

OID

2.5.29.36

严重程度

这个扩展可能是关键或非关键。

B.3.12. policyMappings

Policy Mappings 扩展仅用于 CA 证书。它列出了一个或多个 OID 对，表示一个 CA 的对应策略等同于另一个 CA 的策略。它在跨对证书的上下文中非常有用。

这个扩展可能会被 CA 和应用程序支持。

OID

2.5.29.33

严重程度

这个扩展必须是非关键。

B.3.13. privateKeyUsagePeriod

Private Key Usage Period 扩展允许证书签发者为私钥指定不同于证书本身的有效周期。这个扩展用于数字签名密钥。

注意

PKIX 第 1 部分建议使用这个扩展。符合 PKIX 第 1 的 CA 不得 使用此扩展生成证书。

OID

2.5.29.16

B.3.14. subjectAltName

主题备用名称扩展包括一个或多个替代（非 X.500）名称，供 CA 绑定到认证的公钥。除证书的主题名称或替换证书之外，还可使用该工具。定义的名称表单包括互联网电子邮件地址(SMTP)，如 RFC-822、DNS 名称、IP 地址（IPv4 和 IPv6）以及统一的资源标识符(URI)。

PKIX 要求由主题字段中使用的 X.500 分辨名称(DN)以外的名称表单标识的实体的扩展。PKIX 第 1 部分描述了此扩展与主题字段之间的关系的其他规则。

电子邮件地址可以在 Subject Alternative Name 扩展、证书主题名称字段中或两者中提供。如果电子邮件地址是主题名称的一部分，它的格式必须是 PKCS #9 定义的 EmailAddress 属性。支持 S/MIME 的软件必须能够从主题备用名称扩展或从主题名称字段读取电子邮件地址。

OID

2.5.29.17

严重程度

如果证书的 subject 字段为空，这个扩展必须标记为关键。

B.3.15. subjectDirectoryAttributes

Subject Directory 属性扩展为证书主体提供任何所需的目录属性值。建议不要作为建议 PKIX 标准的重要部分，但可以在本地环境中使用。

OID

2.5.29.9

严重程度

PKIX 第 1 部分要求此扩展标记为非关键。

B.3.16. subjectKeyIdentifier

Subject Key Identifier 扩展标识此证书认证的公钥。这个扩展提供了一种在给定主题名称中可用时区分公钥的方法。

此扩展的值应该通过执行证书 DER-encoded subjectPublicKey 的 SHA 哈希来计算，具体由 PKIX 推荐的。Subject Key Identifier 扩展与 CA 证书的授权密钥标识符扩展结合使用。如果 CA 证书具有 Subject Key Identifier 扩展，则验证证书的授权密钥标识符与 CA 的 Subject Key Identifier 扩展的密钥标识符匹配。在验证器中重新计算此例中的键标识符是不需要的。

PKIX 第 1 部分需要扩展所有 CA 证书，并推荐它用于所有其他证书。

OID

2.5.29.14

严重程度

这个扩展总是非关键。

B.4. CRL 扩展

B.4.1. 关于 CRL 扩展

自初始发布以来，已调整了 CRL 格式的 X.509 标准，以包含 CRL 中的其他信息。此信息通过 CRL 扩展添加。

由 ANSI X9 和 ISO/IEC/ITU 为 X.509 CRL 定义的扩展 [X.509] [X9.55] 允许与 CRL 关联的其他属性。Internet X.509 公共密钥基础架构证书和 CRL 配置集 （通过 RFC 5 280）提供，推荐在 CRL 中使用一组扩展。这些扩展称为 标准 CRL 扩展。

标准还允许创建自定义扩展并包含在 CRL 中。这些扩展称为 私有、专有 或自定义 CRL 扩展，并可包含对机构或业务唯一的信息。应用程序可能无法验证包含私有关键扩展的 CRL，因此不建议在一般上下文中使用自定义扩展。

注意

在 CCITT Recommendations X.208 和 X.209 中，在 CCITT Recommendations X.208 和 X.209 中指定了抽象语法行为(ASN.1)和可辨识(DER)标准。有关 ASN.1 和 DER 的快速摘要，请参阅 A Layman's Guide to a ASN.1、BER 和 DER，它可从 RSA 实验室程序网站 http://www.rsa.com 获得。

B.4.1.1. CRL 扩展结构

CRL 扩展由以下部分组成：

- 扩展的对象标识符(OID)。此标识符唯一标识扩展。它还决定 value 字段中的值的 ASN.1 类型以及值的解释方式。当扩展出现在 CRL 中时，OID 显示为扩展名 ID 字段(extnID)，对应的

ASN.1 编码结构显示为八进制字符串(extnValue)的值；示例显示在 例 B.4 “Pretty-Print Certificate Extensions 示例” 中。

- *标志或布尔值字段名为 critical。*

  分配给此字段的 true 或 false 值指示扩展是否对 CRL 至关重要。

  - *如果扩展至关重要，并且 CRL 发送到不基于扩展 ID 理解扩展的应用程序，则应用程序必须拒绝 CRL。*
  - *如果扩展不重要，并且 CRL 发送到不理解扩展 ID 的扩展的应用程序，则应用程序可以忽略扩展并接受 CRL。*

- *包含扩展名值的 DER 编码的八进制字符串。*

接收 CRL 的应用程序会检查扩展 ID，以确定它是否可以识别 ID。如果能够，它会使用扩展 ID 来确定所使用的值类型。

B.4.1.2. CRL 和 CRL Entry Extensions 示例

以下是 X.509 CRL 版本 2 扩展名示例。CertificateCertificate Systemnbsp;System 可以显示 CRL 可读打印格式，如下所示。如示例所示，CRL 扩展按顺序显示，且每个 CRL 只能有一个特定扩展实例；例如： CRL 只能包含一个授权密钥标识符扩展。但是，CRL-entry 扩展会出现在 CRL 中的相应条目中。

```
Certificate Revocation List:
    Data:
        Version:  v2
        Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
        Issuer: CN=Certificate Authority,O=Example Domain
        This Update: Wednesday, July 29, 2009 8:59:48 AM GMT-08:00
        Next Update: Friday, July 31, 2009 8:59:48 AM GMT-08:00
        Revoked Certificates: 1-3 of 3
            Serial Number: 0x11
            Revocation Date: Thursday, July 23, 2009 10:07:15 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Privilege_Withdrawn
            Serial Number: 0x1A
            Revocation Date: Wednesday, July 29, 2009 8:50:11 AM GMT-08:00
```

```
        Extensions:
            Identifier: Revocation Reason - 2.5.29.21
                Critical: no
                Reason: Certificate_Hold
            Identifier: Invalidity Date - 2.5.29.24
                Critical: no
                Invalidity Date: Sun Jul 26 23:00:00 GMT-08:00 2009
        Serial Number: 0x19
        Revocation Date: Wednesday, July 29, 2009 8:50:49 AM GMT-08:00
        Extensions:
            Identifier: Revocation Reason - 2.5.29.21
                Critical: no
                Reason: Key_Compromise
            Identifier: Invalidity Date - 2.5.29.24
                Critical: no
                Invalidity Date: Fri Jul 24 23:00:00 GMT-08:00 2009
    Extensions:
        Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
            Critical: no
            Access Description:
                Method #0: ocsp
                Location #0: URIName: http://example.com:9180/ca/ocsp
        Identifier: Issuer Alternative Name - 2.5.29.18
            Critical: no
            Issuer Names:
                DNSName: example.com
        Identifier: Authority Key Identifier - 2.5.29.35
            Critical: no
            Key Identifier:
                50:52:0C:AA:22:AC:8A:71:E3:91:0C:C5:77:21:46:9C:
                0F:F8:30:60
        Identifier: Freshest CRL - 2.5.29.46
            Critical: no
            Number of Points: 1
            Point 0
                Distribution Point: [URIName: http://server.example.com:8443/ca/ee/ca/getCRL?
op=getDeltaCRL&crlIssuingPoint=MasterCRL]
        Identifier: CRL Number - 2.5.29.20
            Critical: no
            Number: 39
        Identifier: Issuing Distribution Point - 2.5.29.28
            Critical: yes
            Distribution Point:
                Full Name:
                    URIName: http://example.com:9180/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL
            Only Contains User Certificates: no
            Only Contains CA Certificates: no
            Indirect CRL: no
    Signature:
        Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
        Signature:
            47:D2:CD:C9:E5:F5:9D:56:0A:97:31:F5:D5:F2:51:EB:
            1F:CF:FA:9E:63:D4:80:13:85:E5:D8:27:F0:69:67:B5:
            89:4F:59:5E:69:E4:39:93:61:F2:E3:83:51:0B:68:26:
            CD:99:C4:A2:6C:2B:06:43:35:36:38:07:34:E4:93:80:
```

```
        99:2F:79:FB:76:E8:3D:4C:15:5A:79:4E:E5:3F:7E:FC:
        D8:78:0D:1D:59:A0:4C:14:42:B7:22:92:89:38:3A:4C:
        4A:3A:06:DE:13:74:0E:E9:63:74:D0:2F:46:A1:03:37:
        92:F0:93:D9:AA:F8:13:C5:06:25:02:B0:FD:3B:41:E7:
        62:6F:67:A3:9F:F5:FA:03:41:DA:8D:FD:EA:2F:E3:2B:
        3E:F8:E9:CC:3B:9F:E4:ED:73:F2:9E:B9:54:14:C1:34:
        68:A7:33:8F:AF:38:85:82:40:A2:06:97:3C:B4:88:43:
        7B:AF:5D:87:C4:47:63:4A:11:65:E3:75:55:4D:98:97:
        C2:2E:62:08:A4:04:35:5A:FE:0A:5A:6E:F1:DE:8E:15:
        27:1E:0F:87:33:14:16:2E:57:F7:DC:77:BE:D2:75:AB:
        A9:7C:42:1F:84:6D:40:EC:E7:ED:84:F8:14:16:28:33:
        FD:11:CD:C5:FC:49:B7:7B:39:57:B3:E6:36:E5:CD:B6
```

delta CRL 是 CRL 的子集，仅包含上一次 CRL 发布后的更改。包含 delta CRL 指示器扩展的任何 CRL 都是 delta CRL。

```
ertificate Revocation List:
    Data:
        Version:  v2
        Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
        Issuer: CN=Certificate Authority,O=SjcRedhat Domain
        This Update: Wednesday, July 29, 2009 9:02:28 AM GMT-08:00
        Next Update: Thursday, July 30, 2009 9:02:28 AM GMT-08:00
        Revoked Certificates:
            Serial Number: 0x1A
            Revocation Date: Wednesday, July 29, 2009 9:00:48 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Remove_from_CRL
            Serial Number: 0x17
            Revocation Date: Wednesday, July 29, 2009 9:02:16 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Certificate_Hold
                Identifier: Invalidity Date - 2.5.29.24
                    Critical: no
                    Invalidity Date: Mon Jul 27 23:00:00 GMT-08:00 2009
    Extensions:
        Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
            Critical: no
            Access Description:
                Method #0: ocsp
                Location #0: URIName: http://server.example.com:8443/ca/ocsp
        Identifier: Delta CRL Indicator - 2.5.29.27
            Critical: yes
            Base CRL Number: 39
        Identifier: Issuer Alternative Name - 2.5.29.18
            Critical: no
            Issuer Names:
                DNSName: a-f8.sjc.redhat.com
        Identifier: Authority Key Identifier - 2.5.29.35
            Critical: no
```

```
            Key Identifier:
               50:52:0C:AA:22:AC:8A:71:E3:91:0C:C5:77:21:46:9C:
               0F:F8:30:60
         Identifier: CRL Number - 2.5.29.20
            Critical: no
            Number: 41
         Identifier: Issuing Distribution Point - 2.5.29.28
            Critical: yes
            Distribution Point:
               Full Name:
                  URIName: http://server.example.com:8443/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL
            Only Contains User Certificates: no
            Only Contains CA Certificates: no
            Indirect CRL: no
      Signature:
         Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
         Signature:
            68:28:DA:90:D5:39:CB:6D:BE:42:04:77:C9:E4:09:60:
            C1:97:A6:99:AB:A0:5B:A2:F3:8B:5E:4E:D6:05:70:B0:
            87:1F:D7:0E:4B:C6:B2:DE:8B:92:D8:7C:3B:36:1C:79:
            96:2A:64:E6:7A:25:1D:E7:40:62:48:7A:24:C9:9D:11:
            A6:7F:BB:6B:03:A0:9C:1D:BC:1C:EE:9A:4B:A6:48:2C:
            3B:5E:2B:B1:70:3C:C3:42:96:28:26:AB:82:18:F2:E9:
            F2:55:48:A8:7E:7F:FE:D4:3D:0B:EA:A2:2F:4E:E6:C3:
            C3:C1:6A:E5:C6:85:5B:42:B1:70:2A:C6:E1:D9:0C:AF:
            DA:01:22:FF:80:6E:2E:A7:E5:34:DC:AF:E6:C2:B5:B3:
            1B:FC:28:36:8A:91:4A:22:E7:03:A5:ED:4E:62:0C:D9:
            7F:81:BB:80:99:B8:61:2A:02:C6:9C:41:2E:01:82:21:
            80:82:69:52:BD:B2:AA:DB:0F:80:0A:7E:2A:F3:15:32:
            69:D2:40:0D:39:59:93:75:A2:ED:24:70:FB:EE:19:C0:
            BE:A2:14:36:D0:AC:E8:E2:EE:23:83:DD:BC:DF:38:1A:
            9E:37:AF:E3:50:D9:47:9D:22:7C:36:35:BF:13:2C:16:
            A2:79:CF:05:41:88:8E:B6:A2:4E:B3:48:6D:69:C6:38
```

B.4.2. 标准 X.509 v3 CRL 扩展参考

除了证书扩展外，X.509 还建议标准向 CRL 定义扩展，这提供了将其他属性与互联网 CRL 相关联的方法。它们有两种类型之一：对 CRL 本身进行扩展，并扩展到 CRL 中的单个证书条目。

- 第 B.4.2.1 节 “CRL 的扩展”
- 第 B.4.2.2 节 “CRL Entry Extensions”

B.4.2.1. CRL 的扩展

以下 CRL 描述作为互联网 X.509 v3 公共密钥基础架构的一部分被定义。

- 第 B.4.2.1.1 节 “authorityInfoAccess”
- 第 B.4.2.1.2 节 “authorityKeyIdentifier”
- 第 B.4.2.1.3 节 “CRLNumber”
- 第 B.4.2.1.4 节 “deltaCRLIndicator”
- 第 B.4.2.1.5 节 “FreshestCRL”
- 第 B.4.2.1.6 节 “issuerAltName”
- 第 B.4.2.1.7 节 “issuingDistributionPoint”
- 第 B.4.2.1.5 节 “FreshestCRL”

B.4.2.1.1. authorityInfoAccess

授权信息访问扩展标识如何获取 delta CRL 信息。latestestCRL 扩展放置在完整的 CRL 中，以指示查找最新 delta CRL 的位置。

OID

1.3.6.1.5.5.7.1.1

严重程度

PKIX 要求此扩展必须不是关键。

参数

表 B.39. 授权信息访问配置参数

参数	描述
enable	指定规则是启用还是禁用。默认值为禁用这个扩展。
critical	设定扩展是否标记为关键；默认为非关键。
numberOfAccessDescriptions	表示访问描述的数量，从 0 到任意正整数，默认为 0。 当将此参数设置为 0 以外的整数时，设置数字，然后单击 OK 以关闭该窗口。重新打开该规则的编辑窗口，以及要设置点的字段存在。
accessMethodn	这个参数只接受的值是 calssuers。当可用信息列出了可用于验证 CRL 中的签名的证书时，使用 calssuers 方法。当将 AIA 扩展包含在 CRL 中时，不应使用其他方法。
accessLocationTypen	指定 n 访问描述的访问位置类型。选项可以是 DirectoryName 或 URI。
accessLocationn	如果将 accessLocationType 设为 DirectoryName，则该值必须是 X.500 名称的形式的字符串，类似于证书中的主题名称。例如，CN=CACentral,OU=Research Dept,O=Example Corporation,C=US。 如果将 accessLocationType 设为 URI，则名称必须是 URI；URI 必须是绝对路径名，必须指定主机。例如：http://testCA.example.com/get/crls/here/

B.4.2.1.2. authorityKeyIdentifier

CRL 的授权密钥标识符扩展标识与用于为 CRL 签名的私钥对应的公钥。详情请查看 第 B.3.2 节 “authorityKeyIdentifier” 上的证书扩展讨论。

PKIX 标准建议 CA 必须将其扩展包含在所有 CRL 中，因为 CA 的公钥可能会改变，例如：当密钥被更新时，或者 CA 可能会有多个签名密钥，因为多个并发密钥对或密钥更改。在这些情况下，CA 以多个密钥对结束。在验证证书上的签名时，其他应用程序需要知道签名中使用了哪个密钥。

OID

2.5.29.35

参数

表 B.40. AuthorityKeyIdentifierExt Configuration Parameters

参数	描述
enable	指定规则是启用还是禁用。默认值为禁用这个扩展。
critical	设定扩展是否标记为关键；默认为非关键。

B.4.2.1.3. CRLNumber

CRLNumber 扩展指定 CA 发布的每个 CRL 的后续数字。它允许用户轻松确定特定 CRL 何时取代另一个 CRL。PKIX 要求所有 CRL 都有此扩展。

OID

2.5.29.20

严重程度

这个扩展不一定是关键。

参数

表 B.41. CRLNumber 配置参数

参数	描述
enable	指定规则是否启用，这是默认设置。
critical	设定扩展是否标记为关键; 默认为非关键。

B.4.2.1.4. deltaCRLIndicator

deltaCRLIndicator 扩展会生成 delta CRL，它只包含自上一次 CRL 开始被撤销的证书列表；它还包括对基本 CRL 的引用。这会更新本地数据库，同时忽略了本地数据库中已更改的信息。这可为以 CRL 结构以外的格式存储撤销信息的应用程序显著提高处理时间。

OID

2.5.29.27

严重程度

PKIX 要求此扩展在存在时具有关键性。

参数

表 B.42. DeltaCRL 配置参数

参数	描述
enable	设定是否启用该规则。默认情况下禁用它。
critical	设定扩展是否关键(uncritical)还是非关键。默认情况下，这是关键。

B.4.2.1.5. FreshestCRL

freshestCRL 扩展标识如何获取 delta CRL 信息。latestestCRL 扩展放置在完整的 CRL 中，以指示查找最新 delta CRL 的位置。

OID

2.5.29.46

严重程度

PKIX 要求此扩展必须不是关键。

参数

表 B.43. FreshestCRL 配置参数

参数	*描述*
enable	*设定是否启用扩展规则。默认情况下禁用它。*
critical	*将扩展名标记为关键或非关键。默认为非关键。*
numPoints	*表示 delta CRL 发出点的数量，从 0 到任何正整数，默认值为 0。当将其设置为 0 以外的整数时，设置数字，然后单击 OK 以关闭该窗口。重新打开规则的编辑窗口，以及设置这些点的字段存在。*
pointTypen	*指定 n 点的发布点的类型。对于 numPoint 中指定的每个数字，会有一个相等的 pointType 参数。选项可以是 DirectoryName 或 URIName。*

参数	描述
pointNamen	如果将 pointType 设置为 directoryName，则值必须是以 X.500 名称的形式的字符串，与证书中的主题名称类似。例如，CN=CACentral,OU=Research Dept,O=Example Corporation,C=US。 如果将 pointType 设置为 URIName，则名称必须是 URI；URI 必须是绝对路径名，必须指定主机。例如：http://testCA.example.com/get/crls/here/

B.4.2.1.6. issuerAltName

Issuer Alternative Name 扩展允许其他身份与 CRL 的签发者关联，如绑定属性，如地址、DNS 名称、IP 地址（包括 IPv4 和 IPv6）以及统一的资源指示符(URI)，以及 CRL 签发者。详情请查看 第 B.3.7 节 “issuerAltName 扩展” 上的证书扩展讨论。

OID

2.5.29.18

参数

表 B.44. IssuerAlternativeName 配置参数

参数	描述
enable	设定扩展规则是否启用；默认情况下，这是禁用的。
critical	设定扩展是否至关重要。默认情况下，这不是非常关键。

参数	描述
numNames	设置扩展名中允许的备用名称或身份总数。每个名称都有一组配置参数，nameType 和 name，它必须具有适当的值，或者规则返回错误。通过更改此字段中指定的值来更改身份总数；扩展名中可以包括的身份总数没有限制。各组配置参数可通过从此字段值派生的整数区分。例如，如果 numNames 参数设置为 2，则派生的整数为 0 和 1。
nameTypen	指定 General-name 类型，这可以是以下任意一种： • 如果名称是互联网邮件地址，则 rfc822Name。 • 如果 名称是 X.500 目录名，则名名。 • 如果名称是 DNS 名称，dnsName。 • 如果 名称是 EDI 方名称，则由第三方名称使用。 • 名称是 URI（默认）的 URL。 • 如果名称是 IP 地址，ipaddress。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如：0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:F

参数	描述
	F:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • OID （如果名称是对象标识符）。 • 如果 名称 采用任何其他名称格式，则支持 PrintableString、IA5String、UTF8String、BMPString、Any、和 KerberosName。
namen	指定 general-name 值；允许的值取决于 nameType 字段中指定的名称类型。 • 对于 rfc822Name，该值必须是 local-part@domain 格式的有效互联网邮件地址。 • 对于 directoryName，该值必须是 X.500 名称，与证书中的主题名称类似。例如，CN=CACentral,OU=Research Dept,O=Example Corporation,C=US。 • 对于 dNSName，该值必须是 DNS 格式的有效域名。例如，testCA.example.com。 • 对于 edipartyName，名称必须是 IA5String。例如，Sartor 公司示

参数	描述
	例。 • 对于 URL，值必须是非相对 URI。例如 ： http://testCA.example.com • 对于 iPAddress，该值必须是以点分隔的数字组件表示法指定的有效 IP 地址。它可以是包括子网掩码的 IP 地址或 IP 地址。IPv4 地址的格式必须为 n.n.n 或 n .n.n,m.m.m。例如：128 .21.39.40 或 128.21.39.40,255.255.255.00。IPv6 地址使用 128 位命名空间，使用以冒号分隔的 IPv6 地址以及句点分隔的子网掩码。例如： 0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FFFF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF:FF:FF:FF: FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF: FF • 对于 OID，该值必须是唯一且有效的 OID，在点分隔的数字组件标记中指定。例如，1.2.3.4.55.6.5.99。虽然自定义 OID 可用于评估和测试服务器（在生产环境中），但遵循用于定义 OID 和注册 ID 子树的 ISO 规则。 • 对于其他名称，名称可以是任何其他格式；这支持 PrintableString、IA5String、UTF8String、BMPString、Any、和 KerberosName。PrintableString,IA5String,UTF8String,BMPString, and any set a string to a base-64 编码文件指定子树，如 /var/lib/pki/pki-tomcat/ca/othername.txt。KerberosName 具有 Realm|NameType|NameStrings 格式，如 realm1|0|userID1,userID2。名称必须是文件的绝对路径，该文件必须以 base-64 编码格式包含通用名称。例如： /var/lib/pki/pki-tomcat/ca/extn/ian/othername.txt。

参数	描述

B.4.2.1.7. issuingDistributionPoint

发行的发布点 CRL 扩展可识别特定 CRL 的 CRL 分发点，并指示其定义的撤销类型，如只撤销具有有限原因代码的最终用户证书、CA 证书或撤销具有有限原因代码的证书。

PKIX 第 I 部分不需要此扩展。

OID

2.5.29.28

严重程度

PKIX 要求此扩展在存在时具有关键性。

参数

表 B.45. IssuingDistributionPoint 配置参数

参数	描述
enable	设定是否启用扩展；默认是禁用的。
critical	将扩展名标记为 critical、default 或 noncritical。

参数	描述
pointType	指定从以下内容发出的发行点的类型： • directory name 指定类型是 X.500 目录名称。 • URI 指定类型是统一的资源指示器。 • RelativeToIssuer 指定类型是一个相对可分名称(RDN)，它代表了 DN 的单个节点，如 ou=Engineering。

参数	描述
pointName	给出发布点的名称。发行版点的名称取决于为 pointType 参数指定的值。 • 对于 directoryName，名称必须是 X.500 名称。例如：cn=CRLCentral,ou=Research Dept,o=Example company,c=US。 • 对于 URIName，名称必须是 URI，它是绝对路径名并指定主机。例如：http://testCA.example.com/get/crls/here/  注意 CRL 可以存储在与 CRL 发出点对应的目录条目中，该条目可能与 CA 的目录条目不同。

参数	描述
onlySomeReasons	指定与发布点关联的代码的原因。 Permissible 值是原因代码的组合（未指定的、keyCompromise、cACompromise、ffiliationChanged、it eded、cessationOf Oper、certificateHold 和 removeFromCRL）。如果发行版点包含具有所有原因代码（默认）的已撤销证书，则将该字段留空。
onlyContainsCACerts	指定分发点仅在设置时包含用户证书。默认情况下不设置，这意味着分发点包含所有类型的证书。
indirectCRL	指定分发点包含间接 CRL；默认情况下，不会选择此设置。

B.4.2.2. CRL Entry Extensions

以下部分列出了作为互联网 X.509 v3 公共密钥基础架构的一部分定义的 CRL 条目扩展类型。所有这些扩展都不是关键。

B.4.2.2.1. certificateIssuer

证书发行者扩展标识与间接 CRL 中的条目关联的证书签发者。

这个扩展只与间接 CRL 一起使用，该证书Certificate Systemnbsp;System 不支持。

OID

2.5.29.29

B.4.2.2.2. invalidityDate

Invalidity Date 扩展提供了私钥被入侵或者证书无效的日期。

OID

2.5.29.24

参数

表 B.46. InvalidityDate 配置参数

参数	*描述*
enable	*设定扩展规则是启用还是禁用。默认情况下启用它。*
critical	*将扩展标记为关键或非关键；默认情况下，这不是关键。*

B.4.2.2.3. CRLReason

Reason Code 扩展标识证书撤销的原因。

OID

2.5.29.21

参数

表 B.47. CRLReason 配置参数

参数	*描述*
enable	*设定扩展规则是启用还是禁用。默认情况下启用它。*
critical	*将扩展名标记为关键或非关键。默认情况下，这不关键。*

B.4.3. Netscape-Defined Certificate Extensions 参考

Netscape 为其产品定义了特定的证书扩展。有些扩展现已弃用，其他扩展已被 X.509 建议标准中定义的扩展替代。所有 Netscape 扩展都应标记为非关键，因此证书中存在不会使该证书与其他客户端不兼容。

B.4.3.1. netscape-cert-type

Netscape 证书类型扩展可用于限制可以使用证书的目的。它已被 X.509 v3 扩展 第 B.3.6 节 “extKeyUsage” 和 第 B.3.3 节 “basicConstraints” 替换。

如果证书中存在扩展，它会将证书限制为指定使用。如果扩展不存在，则证书可用于所有应用程序，但对象签名除外。

该值是一个位字符串，其中单个位位置（在设置时）验证用于特定用途的证书，如下所示：

- 位 0：SSL 客户端证书
- 位 1：SSL 服务器证书
- 位 2：S/MIME 证书
- 第 3 位：对象签名证书
- 位 4：保留
- 位 5：SSL CA 证书
- 位 6：S/MIME CA 证书
- 位 7：对象签名 CA 证书

OID

2.16.840.1.113730.1.1

B.4.3.2. netscape-comment

此扩展的值是 IA5String。它是一个在查看证书时向用户显示的注释。

OID

2.16.840.1.113730.13

附录 C. 发布模块参考

使用证书管理器默认配置几个发布器、映射程序和规则模块。

C.1. COMMUNITYLY 插件模块

本节介绍了为证书管理器提供的发布模块。模块供证书管理器用来启用和配置特定的发布程序实例。

C.1.1. FileBasedPublisher

FilebasedPublisher 插件模块配置证书管理器，以将证书和 CRL 发布到文件。此插件可以根据配置发布发布发布器时所选择的复选框，发布 base-64 编码文件、DER 编码文件或两者。可以使用 PrettyPrintCert 和 PrettyPrintCRL 工具转换文件来查看证书和 CRL 内容。有关查看 base-64 和 DER 编码证书和 CRL 中的内容的详情，请参考 第 8.11 节 "查看证书和 CRLs 发布到文件"。

默认情况下，证书管理器不会创建 FileBasedPublisher 模块的实例。

表 C.1. FileBasedPublisher 配置参数

参数	描述
发布程序 ID	指定 publisher 的名称，它是一个没有空格的字母数字字符串。例如，**PublishCertsToFile**。
目录	指定证书管理器创建文件的目录的完整路径 ; 路径可以是绝对路径，或者可以相对于证书证书系统及证书实例目录。例如： **/export/CS/certificates**。

C.1.2. LdapCaCertPublisher

LdapCaCertPublisher 插件模块将证书管理器配置为发布或取消将 CA 证书发布到 CA 目录条目的 caCertificate;binary 属性。

模块将 CA 条目的对象类转换为 pkiCA 或 CertifiedAuthority （如果尚未使用）。同样，如果 CA 没有其他证书，它也会删除 pkiCA 或 certificationAuthority 对象类。

在安装过程中，证书管理器会自动创建 LdapCaCertPublisher 模块的实例，用于将 CA 证书发布到该目录。

表 C.2. LdapCaCertPublisher Configuration Parameters

参数	描述
caCertAttr	指定要发布 CA 证书的 LDAP 目录属性。这必须是 **caCertificate;binary**。
caObjectClass	指定 目录中 CA 条目的对象类。这必须是 **pkiCA** 或 **CertificationAuthority**。

C.1.3. LdapUserCertPublisher

LdapUserCertPublisher 插件模块将证书管理器配置为发布或取消将用户证书发布到 用户

Certificate;binary 属性的用户目录条目。

此模块用于将任何最终用户证书发布到 LDAP 目录。最终用户证书的类型包括 SSL 客户端、S/MIME、SSL 服务器和 OCSP 响应程序。

在安装过程中，证书管理器自动创建一个 LdapUserCertPublisher 模块实例，用于将终端证书发布到该目录。

表 C.3. LdapUserCertPublisher Configuration Parameters

参数	描述
certAttr	指定证书管理器应发布证书的映射条目的目录属性。这必须是 **userCertificate;binary**。

C.1.4. LdapCrlPublisher

LdapCrlPublisher 插件模块将证书管理器配置为发布或取消将 CRL 发布到目录条目的 certificateRevocationList;binary 属性。

在安装过程中，证书管理器自动创建一个 LdapCrlPublisher 模块的实例，用于将 CRL 发布到该目录。

表 C.4. LdapCrlPublisher 配置参数

参数	描述
crlAttr	指定证书管理器应发布 CRL 的映射条目的目录属性。这必须是 **certificateRevocationList;binary**。

C.1.5. LdapDeltaCrlPublisher

LdapDeltaCrlPublisher 插件模块将证书管理器配置为发布或取消将 delta CRL deltaRevocationList 属性发布到目录条目的 deltaRevocationList 属性。

在安装过程中，证书管理器自动创建一个 LdapDeltaCrlPublisher 模块的实例，用于将 CRL 发布到该目录。

表 C.5. LdapDeltaCrlPublisher Configuration Parameters

参数	描述
crlAttr	指定证书管理器应发布 delta CRL 的映射条目的目录属性。这必须是 **deltaRevocationList;binary**。

C.1.6. LdapCertificatePairPublisher

LdapCertificatePairPublisher 插件模块将证书管理器配置为发布或取消将跨签名证书发布到 CA 目录条目的 crossCertPair;binary 属性。

该模块还将 CA 条目的对象类转换为 pkiCA 或 certificationAuthority （如果尚未使用）。同样，如果 CA 没有其他证书，它也会删除 pkiCA 或 certificationAuthority 对象类。

在安装过程中，证书管理器自动创建一个 LdapCertificatePairPublisher 模块实例，名为 LdapCrossCertPairPublisher 将跨证书发布到该目录。

表 C.6. LdapCertificatePairPublisher Parameters

参数	描述
crossCertPairAttr	指定要发布 CA 证书的 LDAP 目录属性。这必须 **跨 CertificatePair;binary**。
caObjectClass	指定 目录中 CA 条目的对象类。这必须是 **pkiCA** 或 **CertificationAuthority**。

C.1.7. OCSPPublisher

OCSPPublisher 插件模块将证书管理器配置为发布其 CRL，并将其 CRL 发布到在线证书 Status Manager。

证书管理器不会在安装时创建 OCSPPublisher 模块的任何实例。

表 C.7. OCSPPublisher 参数

参数	描述
主机	指定在线证书状态管理器的完全限定主机名。
port	指定在线证书状态管理器正在侦听证书管理器的端口号。这是在线证书状态管理器的 SSL 端口号。

参数	描述
path	指定发布 CRL 的路径。这必须是默认路径 **/ocsp/agent/ocsp/addCRL**。
enableClientAuth	设定是否使用客户端（基于证书的）验证来访问 OCSP 服务。
nickname	在 OCSP 服务数据库中指定证书的 nickname 用于客户端身份验证。只有在 **enableClientAuth** 选项被设置为 true 时才使用。

C.2. 映射器插件模块

本节介绍为证书管理器提供的 mapper 插件模块。这些模块将证书管理器配置为启用和配置特定的映射程序实例。

可用的 mapper 插件模块包括：

- *第 C.2.1 节 “LdapCaSimpleMap”*
- *第 C.2.2 节 “LdapDNExactMap”*
- *第 C.2.3 节 “LdapSimpleMap”*
- *第 C.2.4 节 “LdapSubjAttrMap”*
- *第 C.2.5 节 “LdapDNCompsMap”*

C.2.1. LdapCaSimpleMap

LdapCaSimpleMap 插件模块配置了一个证书管理器，以在 LDAP 目录中为 CA 创建条目，然后通过将 CA 的证书映射到目录条目。有关 AVAs 的更多信息，请查看目录文档。

CA 证书映射程序指定是否为 CA 创建条目，将证书映射到现有条目，还是同时执行两者。

如果发布目录中已存在 CA 条目，且分配给此映射程序的 dnPattern 参数的值会被改变，但 uid 和 o 属性相同，mapper 将无法创建第二个 CA 条目。例如，如果目录已经有 uid=CA,ou=Marketing,o=example.com 和 mapper 的 CA 条目，则使用 uid=CA,ou=Engineering,o=example.com 创建另一个 CA 条目。

操作可能会失败，因为目录会将 UID 唯一性插件设置为特定的基本 DN。此设置可防止目录具有该基本 DN 下相同 UID 的两个条目。在本例中，它会阻止目录在 o=example.com 下具有两个条目，它们具有相同的 UID CA。

如果映射程序无法创建第二个 CA 条目，请检查设置 UID 唯一插件的基本 DN，并检查目录中是否存在具有相同 UID 的条目。如有必要，调整 mapper 设置，删除旧 CA 条目，注释掉插件，或者手动创建该条目。

在安装过程中，证书管理器会自动创建 CA 证书映射程序模块的两个实例。映射器的名称如下：

- *CRLs 的 LdapCrlMap （请参阅 第 C.2.1.2 节 “LdapCrlMap”）*
- *LdapCaCertMap for CA 证书（请参阅 第 C.2.1.1 节 “LdapCaCertMap”）。*

表 C.8. LdapCaSimpleMap 配置参数

参数	描述
createCAEntry	创建 CA 的条目（如果选择）（默认）。 如果选中，证书管理器首先会尝试为 目录中的 CA 创建条目。如果在创建条目时证书管理器成功，它将尝试将 CA 的证书发布到该条目。如果未选中此项，则条目必须已存在，才能发布此条目供其发布。

参数	描述
dnPattern	指定证书管理器应使用的 DN 模式来构造用于在发布目录中搜索 CA 的条目。**dnPattern** 的值可以是用逗号分开的 AVAs 列表。AVA 可以是一个变量，如 **cn=$subj.cn**，证书管理器可以从证书主题名称或恒定名称（如 **o=Example Corporation** ）生成。 如果 CA 证书在主题名称中没有 **cn** 组件，请调整 CA 证书映射 DN 模式，以反映 CA 证书正在发布的目录中的 DN。例如，如果 CA 证书 subject DN 是 **o=Example Corporation**，且该目录中的 CA 条目是 **cn=Certificate Authority, o=Example Corporation**，则特征是 **cn=Certificate Authority, o=$subj.o**。 ● 示例 1： **uid=CertMgr, o=Example Corporation** ● 示例 2: **cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US** ● 示例 3： **uid=$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou** 在上面的示例中，**$req** 从证书请求中获取属性，**$subj** 从证书主题名称中获取属性，而 **$ext** 则从证书扩展中获取属性。

C.2.1.1. LdapCaCertMap

LdapCaCertMap mapper 是 LdapCaSimpleMap 模块的实例。证书管理器在安装过程中自动创建此映射程序。

此映射映射程序在 目录中为 CA 创建一个条目，并将 CA 证书映射到 目录中的 CA 条目。

默认情况下，映射程序配置为在 目录中为 CA 创建条目，用于查找 CA 条目的默认 DN 模式如下：

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.1.2. LdapCrlMap

LdapCrlMap mapper 是 LdapCaSimpleMap 模块的实例。证书管理器在安装过程中自动创建此映射程序。

此映射映射程序在 目录中为 CA 创建一个条目，并将 CRL 映射到 目录中的 CA 条目。

默认情况下，映射程序配置为在 目录中为 CA 创建条目。查找 CA 条目的默认 DN 模式如下：

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.2. LdapDNExactMap

LdapDNExactMap 插件模块配置一个证书管理器，通过搜索与证书主题名称匹配的 LDAP 条目 DN 将证书映射到 LDAP 目录条目。要使用此映射程序，每个证书主题名称必须与目录条目中的 DN 完全匹配。例如，当搜索该条目目录时，如果证书主题名称为 uid=jdoe、o=Example Corporation、c=US，则证书管理器仅搜索具有 DN uid=jdoe、o=Example Corporation、c=US.

如果未找到匹配的条目，服务器会返回错误，不会发布证书。

这个映射程序不需要任何参数的值，因为它从证书获取所有值。

C.2.3. LdapSimpleMap

LdapSimpleMap 插件模块配置一个证书管理器，通过分离证书请求中指定的组件的 DN，将证书映射到 LDAP 目录条目。有关 AVAs 的更多信息，请参阅目录文档。

默认情况下，证书管理器使用基于简单映射器的映射程序规则。在安装过程中，证书管理器自动创建一个简单映射程序模块的实例，名为 LdapUserCertMap。默认映射程序将各种最终用户证书映射到其对应的目录条目。

简单映射程序需要一个参数 dnPattern。dnPattern 的值可以是用逗号分开的 AVAs 列表。AVA 可以是变量，如 uid=$subj.UID 或一个恒定（如 o=Example Corporation ）。

- 示例 1： uid=CertMgr, o=Example Corporation
- 示例 2: cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US
- 示例 3：uid=$req.HTTP_PARAMS.uid,

e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou

在示例中，$req 获取证书请求的属性，$subj 从证书主题名称中获取属性，而 $ext 则从证书扩展中获取属性。

C.2.4. LdapSubjAttrMap

LdapSubjAttrMap 插件模块配置证书管理器，以使用可配置 LDAP 属性将证书映射到 LDAP 目录条目。要使用此映射程序，目录条目必须包含指定的 LDAP 属性。

此映射程序需要主题 DN 的确切模式，因为证书管理器在目录中搜索包含与整个主题 DN 完全匹配的属性。例如，如果指定的 LDAP 属性是 certSubjectDN，且证书主题名称为 uid=jdoe, o=Example Corporation, c=US，则证书管理器将搜索包含属性 certSubjectDN=uid=jdoe、o=Example Company, c=US 的条目。

如果没有找到匹配的条目，服务器会返回错误并将其写入日志。

表 C.9 “LdapSubjAttrMap Parameters” 描述这些参数。

表 C.9. LdapSubjAttrMap Parameters

参数	描述
certSubjNameAttr	指定包含证书主题名称的 LDAP 属性的名称作为其值。默认为 **certSubjectName**，但可以配置为任何 LDAP 属性。
searchBase	指定用于启动属性搜索的基础 DN。permissible 值是 LDAP 条目的有效 DN，如 **o=example.com、c=US**。

C.2.5. LdapDNCompsMap

LdapDNCompsMap 插件模块实施 DN 组件映射程序。此映射映射通过从组件构建条目到 LDAP 目录条目来映射证书，如 cn、ou、o 和 c，在证书主题名称中指定的，然后在证书主题名称中指定，然后将其用作搜索 DN 在 目录中定位条目。mapper 查找以下条目：

- *CA 的条目位于 目录中，以发布 CA 证书和 CRL。*

- 目录中用于发布终止证书的端点条目。

mapper 使用 DN 组件来构建搜索 DN。mapper 还使用一个可选的根搜索 DN。服务器使用 DN 组件组成 LDAP 条目，以开始子树搜索和过滤器组件组成子树的搜索过滤器。如果不配置 DN 组件，服务器将为子树使用基本 DN。如果基础 DN 为空，且任何 DN 组件都不匹配，则返回一个错误。如果 DN 组件和过滤组件都不匹配，则返回错误。如果过滤器组件是 null，则执行基本搜索。

DNComps 和 filterComps 参数接受用逗号分开的有效 DN 组件或属性。参数不接受多个属性的条目；例如，filterComps 可以设置为 cn，但不要设置为 cn,ou 2,ou1。要使用同一属性的多个实例创建过滤器，如如果目录条目包含多个内容，请修改 LdapDNCompsMap 模块的源代码。

以下组件通常用于 DN：

- UID 代表目录中用户的用户 ID。
- cn 代表 目录中用户的通用名称。
- ou 代表目录中的一个机构单元。
- O 代表目录中的组织。
- l 代表本地性(city)。
- st 代表状态。
- c 表示国家/地区。

例如，以下 DN 代表一个名为 Jane Doe 的用户，它适用于 Example Corporation 的销售部门，该部门位于加利福尼亚州的 Mountain View, California：

```
cn=Jane Doe, ou=Sales, o=Example Corporation, l=Mountain View, st=California, c=US
```

证书管理器可以使用部分或全部组件（cn、ou、o、k、和 c）来构建用于搜索目录的 DN。 在创建映射程序规则时，可为服务器指定这些组件以构建 DN；即，组件与目录中的属性匹配。这通过 dnComps 参数进行设置。

例如，组件 cn、ou、o 和 c 被设置为 dnComps 参数的值。要在 目录中找到 Jane Doe 的条目，证书管理器通过读取证书的 DN 属性值来构造以下 DN，并使用 DN 作为搜索目录的基础：

```
cn=Jane Doe, ou=Sales, o=Example Corporation, c=US
```

- 主题名称不需要具有 dnComps 参数中指定的所有组件。服务器会忽略任何不属于主体名称的组件，如 l 和 st。

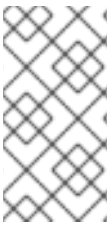
- 未指定的组件不会用于构建 DN。在示例中，如果 ou 组件没有包括，服务器使用此 DN 作为搜索目录的基础：

  ```
  cn=Jane Doe, o=Example Corporation, c=US
  ```

对于 dnComps 参数，输入这些 DN 组件，供证书管理器用来完全用来形成 LDAP DN。但在某些情况下，证书中的主题名称可能与目录中的多个条目匹配。然后，证书管理器可能无法从 DN 中获取不同的匹配条目。例如，Subject name cn=Jane Doe, ou= sales, o=Example Corporation, c=US 可能与目录中名为 Jane Doe 的两个用户匹配。如果出现这种情况，证书管理器需要额外的条件来决定哪个条目与证书的主题对应。

要指定证书管理器要使用的组件来区分目录中的不同条目，请使用 filterComps 参数；详情请查看 表 C.10 “LdapDNCompsMap 配置参数”。例如，如果 cn、ou、o 和 c 是 dnComps 参数的值，则只有使用 l 属性来区分与相同 cn、ou、o、o 和 c 值相同的条目。

如果两个 Jane Doe 条目通过 uid 属性的值区分，那么一个条目的 uid 是 janedoe1，而另一个条目的 uid 是 janedoe2 - 可将证书的主题名称设置为包含 uid 组件。



注意

e、l 和 st 组件不包括在为终端实体提供的标准证书请求表单集中。这些组件可以在表单中添加，或者在编辑证书中的主题名称时，需要发出代理来插入这些组件。

C.2.5.1. LdapDNCompsMap 的配置参数

使用这个配置时，证书管理器使用 dnComps 值将证书映射到 LDAP 目录中的证书，以形成 DN 和过滤器Comps 值组成子树的搜索过滤器。

- *如果格式 DN 为 null，服务器会使用子树的 baseDN 值。如果格式的 DN 和基础 DN 为 null，服务器会记录错误。*

- *如果过滤器是 null，服务器会使用 baseDN 值来搜索。如果过滤器和基础 DN 为 null，服务器会记录错误。*

表 C.10 “LdapDNCompsMap 配置参数” 描述这些参数。

表 C.10. LdapDNCompsMap 配置参数

参数	描述
baseDN	指定开始搜索发布目录中条目的 DN。如果 **dnComps** 字段为空，服务器使用基础 DN 值在 目录中开始其搜索。
dnComps	指定发布目录中的证书管理器应开始搜索与 CA 或终端用户信息匹配的 LDAP 条目。 例如，如果 **dnComps** 使用 DN 的 **o** 和 **c** 属性，服务器会从 目录中的 **o=**机构、**c=** *country* 条目开始搜索，其中机构和国家/地区的值替换为证书中的 DN 的值。 *如果 **dnComps** 字段为空，服务器会检查 **baseDN** 字段，搜索由该 DN 指定的条目与 **filterComps** 参数值指定的过滤器指定的目录树。* *允许的数值是有效的 DN 组件或以逗号分开的属性。*
filterComps	指定证书管理器应使用的组件来过滤搜索结果中的条目。服务器使用 **filterComps** 值组成子树的 LDAP 搜索过滤器。服务器通过从证书主题名称收集这些属性的值来构建过滤器；它使用过滤器搜索和匹配 LDAP 目录中的条目。 如果服务器在 目录中找到多个与从证书收集的信息匹配的条目，则搜索将成功，并且服务器可以选择性地执行验证。例如，如果 **filterComps** 设为使用电子邮件和用户 ID 属性(**filterComps=e,uid**)，服务器会在目录中搜索与从证书收集的信息匹配的条目。 允许的数值是用逗号分隔的证书 DN 中的有效目录属性。过滤器的属性名称需要是来自证书的属性名称，而不是从 LDAP 目录中的一个属性名称。例如，大多数证书都有一个用于用户电子邮件地址的 **e** 属性；LDAP 调用该属性 **mail**。

C.3. 规则实例

本节讨论已设置的规则实例。

C.3.1. LdapCaCertRule

LdapCaCertRule 可用于将 CA 证书发布到 LDAP 目录中。

表 C.11. LdapCaCert Rule 配置参数

参数	值	描述
type	**cacert**	指定将要发布的证书类型。
predicate		为发布器指定 predicate。
enable	**是**	启用规则。
mapper	**LdapCaCertMap**	指定规则中使用的映射程序。有关映射程序的详情，请查看 第 C.2.1.1 节 “LdapCaCertMap”。
publisher	**LdapCaCertPublisher**	指定规则中使用的 publisher。如需有关发布程序的详细信息，请参阅 第 C.1.2 节 “LdapCaCertPublisher”。

C.3.2. LdapXCertRule

LdapXCertRule 用于向 LDAP 目录发布跨对证书。

表 C.12. LdapXCert Rule 配置参数

参数	值	描述
type	**xcert**	指定将要发布的证书类型。
predicate		为发布器指定 predicate。
enable	**是**	启用规则。
mapper	**LdapCaCertMap**	指定规则中使用的映射程序。有关映射程序的详情，请查看 第 C.2.1.1 节 “LdapCaCertMap”。

参数	值	描述
publisher	**LdapCrossCertPairPublisher**	指定规则中使用的 publisher。如需了解有关此发布者的详细信息，请参阅 第 C.1.6 节 “LdapCertificatePairPublisher”。

C.3.3. LdapUserCertRule

LdapUserCertRule 用于向 LDAP 目录发布用户证书。

表 C.13. LdapUserCert Rule 配置参数

参数	值	描述
type	证书	指定将要发布的证书类型。
predicate		为发布器指定 predicate。
enable	**是**	启用规则。
mapper	**LdapUserCertMap**	指定规则中使用的映射程序。有关映射程序的详情，请查看 第 C.2.3 节 “LdapSimpleMap”。
publisher	**LdapUserCertPublisher**	指定规则中使用的 publisher。如需有关发布程序的详细信息，请参阅 第 C.1.3 节 “LdapUserCertPublisher”。

C.3.4. LdapCRLRule

LdapCRLRule 用于将 CRL 发布到 LDAP 目录。

表 C.14. LdapCRL Rule 配置参数

参数	值	描述
type	**crl**	指定将要发布的证书类型。
predicate		为发布器指定 predicate。
enable	**是**	启用规则。

参数	值	描述
mapper	**LdapCrlMap**	指定规则中使用的映射程序。有关映射程序的详情，请查看 第 C.2.1.2 节 “LdapCrlMap”。
publisher	**LdapCrlPublisher**	指定规则中使用的 publisher。如需有关发布程序的详细信息，请参阅 第 C.1.4 节 “LdapCrlPublisher”。

附录 D. ACL 参考

本节介绍每个资源控制的内容，列出描述这些操作结果的可能操作，并为定义的每个 ACL 资源提供默认 ACl。每个子系统仅包含与该子系统相关的 ACL。

D.1. 关于 ACL 配置文件

访问控制 是设置能够访问服务器一部分的规则以及用户可以执行的操作的方法。依赖于 LDAP 目录服务的四个子系统，并使用 Java 控制台 - CA、KRA、OCSP 和 TKS - 所有实施 LDAP 风格的访问控制来访问其资源。这些访问控制列表(ACL)位于 /var/lib/pki/instance_name/conf/子系统/acl.ldif 文件中。

注意

本节只提供有关访问控制概念的简单概述。《Red Hat Directory Server 管理指南》中的管理访问控制章节中提供了更详细的信息。
https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/10/html/Administration_Guide/Managing_Access_Control.html

CertificateCertificate Systemnbsp;System ACL 文件是内部数据库载入的 LDIF 文件。单个 ACL 定义为 resourceACLS 属性，它们识别受保护子系统的区域，然后是正在设置的所有特定访问控制的列表。

```
resourceACLS: class_name:all rights: allow|deny (rights) type=target description
```

允许或拒绝访问资源的每个规则都称为访问控制指令(ACI)。（资源的所有 ACI 的总和是一个访问控制列表。） 在定义实际 ACI 之前，ACL 属性首先应用于证书证书系统nbsp;System 子系统使用的特定插件类。这会将每个 ACL 集中到子系统执行的特定功能，为实例提供更多安全性，并更好地控制应用的 ACL。

例 D.1. 列出证书配置集的默认 ACL

```
resourceACLS: certServer.ca.profiles:list:allow (list) group="Certificate Manager Agents":Certificate Manager agents may list profiles
```

因为每个子系统（CA、KRA、OCSP 和 TKS）都有不同的资源用于其操作，所以每个子系统实例都有自己的 acl.ldif 文件，自己的 ACL。

每个 ACI 均可以定义什么访问或行为（右侧）以及 ACI 应用到谁（目标）。ACI 的基本格式为：

```
allow|deny (rights) user|group
```

权限 是 ACI 允许用户执行的操作类型。对于 LDAP ACI，目录条目的权限相对有限，如搜索、读取、写入和删除等。CertificateCertificate Systemnbsp;System 使用其他权限，它们涵盖常见的 PKI 任务，如撤销、提交和分配。

如果在 ACI 中未明确允许某一操作，则它会被隐式拒绝。如果一个 ACI 中明确拒绝某个操作，则它会显示显式允许的任何 ACI。拒绝规则始终表现为允许规则提供额外的安全性。

每个 ACI 必须应用到特定的用户或组。这使用几个常用条件（通常是 user= 或 group= ）进行设置，尽管存在其它选项（如 ipaddress= ），后者定义了基于客户端的访问而不是基于条目的访问。如果有多个条件，则可以使用双线(||)运算符组成条件，使用双线(||)运算符符号表示逻辑退出（"或"），以及双符号(&&)运算符，使用符号表示逻辑（"和"）。例如，group="group1" || "group2 "。

资源ACLS 属性值 的每个区域都在 表 D.1 “ACL 属性值的部分” 中定义。

表 D.1. ACL 属性值的部分

值	描述
class_name	将 ACI 应用到的插件类。
全部操作	ACI 定义中涵盖的每个操作列表。单一 ACI 和多个 ACI 在单一 *资源ACLS* 属性中可以有多个操作。
allow|deny	目标用户或组是否允许操作。
(*operations*)	允许或拒绝的操作。
type=target	用于标识此适用于谁的目标。这通常是用户（如 **user=***"name"*）或组(**group=***"group"*)。如果有多个条件，则可以使用双 pipe(||)运算符（逻辑"或"）和双倍(&&)运算符（逻辑"和"）组成条件。例如，**group="group1" || "group2** "。
description	有关 ACL 用途的描述。

D.2. 常见 ACL

本节介绍所有四个子系统类型通用的默认访问控制配置。这些访问控制规则管理对基本和常见配置设置的访问，如记录和添加用户和组。

重要

这些 ACL 常用于每个子系统实例的 acl.ldif 文件中出现相同的 ACL。这些不是 共享 ACL，这意味着所有子系统实例都会保留配置文件或设置。与所有其他实例配置一样，这些 ACL 独立于其他子系统实例在实例特定 acl.ldif 文件中进行维护。

D.2.1. certServer.acl.configuration

控制 ACL 配置的操作。默认配置是：

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表 D.2. certServer.acl.configuration ACL 总结

操作	描述	allow/Deny Access	目标用户/组
读取	查看 ACL 资源并列出 ACL 资源、ACL 列出 evaluator 和 ACL 评估器类型。	Allow	管理员 代理 审核员
modify	添加、删除和更新 ACL 评估器。	Allow	管理员

D.2.2. certServer.admin.certificate

控制哪些用户可以通过证书管理器导入证书。默认情况下，此操作允许任何人使用。默认配置是：

```
allow (import) user="anybody"
```

注意

此条目与用于配置实例的 CA 管理 Web 界面相关联。这个 ACL 只在实例配置过程中可用，且在 CA 运行后不可用。

表 D.3. certServer.admin.certificate ACL Summary

操作	描述	allow/Deny Access	目标用户/组
import	导入 CA 管理员证书，并使用序列号检索证书。	Allow	任何人

D.2.3. certServer.auth.configuration

控制身份验证配置中的操作。

> *allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators*

表 D.4. certServer.auth.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看身份验证插件、身份验证类型、配置的身份验证管理器插件和身份验证实例。列出身份验证管理器插件和验证管理器实例。	Allow	管理员 代理 审核员
modify	添加或删除身份验证插件和身份验证实例。修改身份验证实例。	Allow	管理员

D.2.4. certServer.clone.configuration

控制谁可以读取和修改克隆中使用的配置信息。默认设置为：

> *allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators"*

表 D.5. certServer.clone.configuration ACL 总结

操作	描述	allow/Deny Access	目标用户/组
读取	查看原始实例配置。	Allow	企业管理员
modify	修改原始实例配置。	Allow	企业管理员

D.2.5. certServer.general.configuration

控制子系统实例的常规配置的访问权限，包括谁可以查看和编辑 CA 的设置。

> *allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";allow (modify) group="Administrators"*

表 D.6. certServer.general.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看操作环境、LDAP 配置、SMTP 配置、服务器统计数据、加密、令牌名称、证书名称、证书别名、服务器、CA 证书以及所有管理证书。	Allow	管理员 代理 审核员
modify	修改 LDAP 数据库、SMTP 和加密的设置。签发导入证书、安装证书、信任和不信任 CA 证书、导入跨对证书以及删除证书。执行服务器重启和停止操作。登录所有令牌并检查令牌状态。根据需要运行自我证明。获取证书信息。处理证书主题名称。验证证书主题名称、证书密钥长度和证书扩展。	Allow	管理员

D.2.6. certServer.log.configuration

控制证书管理器的日志配置的访问权限，包括更改日志设置。

> *allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";allow (modify) group="Administrators"*

表 D.7. certServer.log.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组

操作	描述	allow/Deny Access	目标用户/组
读取	查看日志插件信息、插件配置和日志实例配置。列出日志插件和日志实例（不包括 NTEventLog）。	Allow	管理员 代理 审核员
modify	添加和删除日志插件和日志实例。修改日志实例，包括日志滚动参数和日志级别。	Allow	管理员

D.2.7. certServer.log.configuration.fileName

限制对访问来更改实例日志的文件名。

> *allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";deny (modify) user=anybody*

表 D.8. certServer.log.configuration.fileName ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看日志实例的 **fileName** 参数的值。	Allow	管理员 代理 审核员
modify	更改日志实例的 **fileName** 参数的值。	deny	任何人

D.2.8. certServer.log.content.system

控制谁可以查看实例的日志。

> *allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors"*

表 D.9. certServer.log.content.system ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看日志内容。列出所有日志。	Allow	管理员 代理 审核员

D.2.9. certServer.log.content.transactions

控制谁可以查看实例的事务日志。

> *allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors"*

表 D.10. certServer.log.content.transactions ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看日志内容。列出所有日志。	Allow	管理员 代理 审核员

D.2.10. certServer.log.content.signedAudit

控制谁有权访问签名的审计日志。默认设置为：

> *allow (read) group="Auditors"*

表 D.11. certServer.log.content.signedAudit ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看日志内容。列出日志。	Allow	审核员

D.2.11. certServer.registry.configuration

控制管理 registry 的访问，用于注册插件模块的文件。目前，这仅用于注册证书配置集插件。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表 D.12. certServer.registry.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看管理 registry、支持的策略约束、配置集插件配置和配置集插件列表。	Allow	管理员 代理 审核员
modify	注册单个配置集实施插件。	Allow	管理员

D.3. 特定于证书的 ACL

本节涵盖为证书管理器设置的默认访问控制配置属性。CA ACL 配置还包括 第 D.2 节 “常见 ACL” 中列出的所有常见 ACL。

每个 CA 接口（管理控制台和代理及端点服务页面）以及列出和下载证书等常见操作都有访问控制规则。

D.3.1. certServer.admin.ocsp

将证书管理器的 OCSP 配置的访问权限限制为企业 OCSP 管理员组的成员。

```
allow (modify,read) group="Enterprise OCSP Administrators"
```

表 D.13. certServer.admin.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	修改 OCSP 配置，OCSP 存储配置以及默认的 OCSP 存储。	Allow	Enterprise OCSP 管理员

操作	描述	allow/Deny Access	目标用户/组
读取	阅读 OCSP 配置。	Allow	Enterprise OCSP 管理员

D.3.2. certServer.ca.certificate

控制代理服务界面中证书的基本管理操作，包括导入和撤销证书。默认配置是：

```
allow (import,unrevoke,revoke,read) group="Certificate Manager Agents"
```

表 D.14. certServer.ca.certificate ACL Summary

操作	描述	allow/Deny Access	目标用户/组
import	按序列号检索证书。	Allow	证书管理器代理
unrevoke	更改证书的状态。	Allow	证书管理器代理
revoke	更改要撤销的证书的状态。	Allow	证书管理器代理
读取	根据请求 ID 检索证书，并根据请求 ID 或序列号显示证书详情。	Allow	证书管理器代理

D.3.3. certServer.ca.certificates

控制通过代理服务接口列出或撤销证书的操作。默认配置是：

```
allow (revoke,list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

表 D.15. certServer.ca.certificates ACL Summary

操作	描述	allow/Deny Access	目标用户/组
revoke	撤销证书，或批准证书撤销请求。从 TPS 撤销证书。提示用户输入与撤销请求相关的附加数据。	Allow	证书管理器代理 注册管理器代理
list	根据搜索列出证书。根据一系列序列号，检索有关证书范围的详细信息。	Allow	证书管理器代理 注册管理器代理

操作	描述	allow/Deny Access	目标用户/组

D.3.4. certServer.ca.configuration

控制证书管理器的一般配置操作。默认配置是：

> *allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"*

表 D.16. certServer.ca.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看 CRL 插件信息、常规 CA 配置、CA 连接器配置、CRL 配置集配置、请求通知配置、重新调用通知配置、队列通知配置和 CRL 扩展配置。列出 CRL 扩展配置和 CRL 发出点配置。	Allow	管理员 代理 审核员
modify	添加和删除 CRL 发出点。修改常规 CA 设置、CA 连接器配置、CRL 发出点配置、CRL 配置、请求通知配置、重新调用通知配置、队列通知配置和 CRL 扩展配置。	Allow	管理员

D.3.5. certServer.ca.connector

控制通过特殊连接器向 CA 提交请求的操作。默认配置是：

> *allow (submit) group="Trusted Managers"*

表 D.17. certServer.ca.connector ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
submit	提交来自远程可信管理者的请求。	Allow	可信管理器

D.3.6. certServer.ca.connectorInfo

控制对连接器信息的访问，以管理 CA 和 KRA 之间的可信关系。这些信任关系是特殊的配置，允许 CA 和 KRA 自动连接执行密钥归档和恢复操作。这些信任关系是通过特殊的连接器插件配置。

```
allow (read) group="Enterprise KRA Administrators";allow (modify) group="Enterprise KRA Administrators" || group="Subsystem Group"
```

表 D.18. certServer.ca.connectorInfo ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	读取连接器插件设置。	Allow	企业 KRA 管理员
modify	修改连接器插件设置。	Allow	企业 KRA 管理员 子系统组

D.3.7. certServer.ca.crl

通过代理服务接口控制读取或更新 CRL 的访问。默认设置为：

```
allow (read,update) group="Certificate Manager Agents"
```

表 D.19. certServer.ca.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	显示 CRL 并获取 CA CRL 处理的详细信息。	Allow	证书管理器代理
update	更新 CRL。	Allow	证书管理器代理

D.3.8. certServer.ca.directory

控制对用于发布证书和 CRL 的 LDAP 目录的访问。

allow (update) group="Certificate Manager Agents"

表 D.20. certServer.ca.directory ACL Summary

操作	描述	allow/Deny Access	目标用户/组
update	将 CA 证书、CRL 和用户证书发布到 LDAP 目录。	Allow	证书管理器代理

D.3.9. certServer.ca.group

控制为证书管理器实例添加用户和组的内部数据库的访问权限。

allow (modify,read) group="Administrators"

表 D.21. certServer.ca.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	为实例创建、编辑或删除用户和组条目。在属性中添加或修改用户证书	Allow	管理员
读取	查看实例的用户和组条目。	Allow	管理员

D.3.10. certServer.ca.ocsp

通过代理服务界面控制访问和读取 OCSP 信息的功能，如用量统计。

allow (read) group="Certificate Manager Agents"

表 D.22. certServer.ca.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	检索 OCSP 用量统计。	Allow	证书管理器代理

D.3.11. certServer.ca.profile

控制代理服务页面中对证书配置集配置的访问。

> *allow (read,approve) group="Certificate Manager Agents"*

表 D.23. certServer.ca.profile ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看证书配置集的详细信息。	Allow	证书管理器代理
批准	批准并启用证书配置集。	Allow	证书管理器代理

D.3.12. certServer.ca.profiles

控制列出代理服务接口中的证书配置集的访问。

> *allow (list) group="Certificate Manager Agents"*

表 D.24. certServer.ca.profiles ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
list	列出证书配置集。	Allow	证书管理器代理

D.3.13. certServer.ca.registerUser

定义哪个组或用户可为实例创建代理用户。默认配置是：

> *allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" || group="Enterprise TPS Administrators"*

表 D.25. certServer.ca.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	注册新代理。	Allow	企业管理员
读取	读取现有代理信息。	Allow	企业管理员

D.3.14. certServer.ca.request.enrollment

控制如何处理和分配注册请求。默认设置为：

> *allow (submit) user="anybody";allow (read,execute,assign,unassign) group="Certificate Manager Agents"*

表 D.26. certServer.ca.request.enrollment ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看注册请求。	Allow	证书管理器代理
execute	修改请求的批准状态。	Allow	证书管理器代理
submit	论坛请求.	Allow	任何人
分配	为证书管理器代理分配一个请求。	Allow	证书管理器代理
unassign	更改请求的分配。	Allow	证书管理器代理

D.3.15. certServer.ca.request.profile

控制基于证书配置文件的请求的处理。默认设置为：

> *allow (approve,read) group="Certificate Manager Agents"*

表 D.27. certServer.ca.request.profile ACL 概述

操作	描述	allow/Deny Access	目标用户/组
批准	修改基于证书配置集的证书请求的批准状态。	Allow	证书管理器代理
读取	查看基于证书配置文件的证书请求。	Allow	证书管理器代理

D.3.16. certServer.ca.requests

控制可以列出代理服务接口中的证书请求。

> *allow (list) group="Certificate Manager Agents"|| group="Registration Manager Agents"*

表 D.28. certServer.ca.requests ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	检索请求范围的详细信息，并使用复杂过滤器搜索证书。	Allow	证书管理器代理 注册管理器代理

D.3.17. certServer.ca.systemstatus

控制谁可以查看证书管理器实例的统计信息。

```
allow (read) group="Certificate Manager Agents"
```

表 D.29. certServer.ca.systemstatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看统计数据。	Allow	证书管理器代理

D.3.18. certServer.ee.certchain

控制在端点页面中可以访问 CA 证书链的人员。

```
allow (download,read) user="anybody"
```

表 D.30. certServer.ee.certchain ACL Summary

操作	描述	allow/Deny Access	目标用户/组
下载	下载 CA 的证书链。	Allow	任何人
读取	查看 CA 的证书链。	Allow	任何人

D.3.19. certServer.ee.certificate

通过端点页面，控制谁可以访问证书，例如导入或撤销证书。

```
allow (renew,revoke,read,import) user="anybody"
```

表 D.31. certServer.ee.certificate ACL Summary

操作	描述	allow/Deny Access	目标用户/组
续订	提交续订现有证书的请求。	Allow	任何人
revoke	提交用户证书的撤销请求。	Allow	任何人
读取	根据证书序列号或请求 ID 检索和查看证书。	Allow	任何人
import	根据序列号导入证书。	Allow	任何人

D.3.20. certServer.ee.certificates

控制可以列出已撤销的证书或在端点页面中提交撤销请求。

```
allow (revoke,list) user="anybody"
```

表 D.32. certServer.ee.certificates ACL Summary

操作	描述	allow/Deny Access	目标用户/组
revoke	提交需要撤销的证书列表。	Allow	要撤销的证书的主题必须与向 CA 进行身份验证的证书匹配。
list	搜索与指定条件匹配的证书。	Allow	任何人

D.3.21. certServer.ee.crl

通过最终用户页面控制对 CRL 的访问。

```
allow (read,add) user="anybody"
```

表 D.33. certServer.ee.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	检索并查看证书撤销列表。	Allow	任何人
add	将 CRL 添加到 OCSP 服务器中。	Allow	任何人

D.3.22. certServer.ee.profile

控制终端页面中证书配置集的一些访问，包括谁可以查看配置集的详情或通过配置集提交请求。

```
allow (submit,read) user="anybody"
```

表 D.34. certServer.ee.profile ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	通过证书配置文件提交证书请求。	Allow	任何人
读取	显示证书配置文件的详细信息.	Allow	任何人

D.3.23. certServer.ee.profiles

控制可以在端点页面中列出活跃的证书配置集。

```
allow (list) user="anybody"
```

表 D.35. certServer.ee.profiles ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	列出证书配置集。	Allow	任何人

D.3.24. certServer.ee.request.ocsp

根据客户端提交 OCSP 请求的 IP 地址控制访问。

```
allow (submit) ipaddress=".*"
```

表 D.36. certServer.ee.request.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	提交 OCSP 请求。	Allow	所有 IP 地址

D.3.25. certServer.ee.request.revocation

控制用户在端点页面中提交证书撤销请求。

```
allow (submit) user="anybody"
```

表 D.37. certServer.ee.request.revocation ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	提交撤销证书的请求。	Allow	任何人

D.3.26. certServer.ee.requestStatus

控制谁可以在端点页面中查看证书请求的状态。

```
allow (read) user="anybody"
```

表 D.38. certServer.ee.requestStatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	检索已针对该请求发布的任何证书的请求和序列号。	Allow	任何人

D.3.27. certServer.job.configuration

控制谁可以为证书管理器配置作业。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表 D.39. certServer.job.configuration ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看基本作业设置、作业实例设置和作业插件设置。列出作业插件和作业实例。	Allow	管理员 代理 审核员

操作	描述	allow/Deny Access	目标用户/组
modify	添加和删除作业插件和作业实例。修改作业插件和作业实例。	Allow	管理员

D.3.28. certServer.profile.configuration

控制对证书配置集配置的访问。默认设置为：

> *allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"*

表 D.40. certServer.profile.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看证书配置集默认值和约束、输入、输出、输出配置、默认配置、策略约束配置和证书配置集实例配置。列出证书配置集插件和证书配置集实例。	Allow	管理员 代理 审核员
modify	添加、修改和删除证书配置集默认值和约束、输入、输出和证书配置集实例。添加和修改默认策略限制配置。	Allow	管理员

D.3.29. certServer.publisher.configuration

控制谁可以查看并编辑证书管理器发布配置。默认配置是：

> *allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";allow (modify) group="Administrators"*

表 D.41. certServer.publisher.configuration ACL Summary

操作	描述	allow/Deny Access	目标用户/组

操作	描述	allow/Deny Access	目标用户/组
读取	查看 LDAP 服务器目的地信息、发布程序插件配置、发布程序实例配置、映射程序插件配置、映射程序实例配置、规则插件配置和规则实例配置。列出发布程序插件和实例、规则插件和实例，以及程序插件和实例。	Allow	管理员 代理 审核员
modify	添加和删除发布程序插件、发布程序实例、映射程序插件、映射程序实例、规则插件和规则实例。修改发布程序实例、映射程序实例、规则实例和 LDAP 服务器目的地信息。	Allow	管理员

D.3.30. certServer.securitydomain.domainxml

控制由域主机证书管理器在 registry 中维护的安全域信息的访问。安全域配置可以在配置过程中直接被子系统实例访问和修改，因此必须始终允许对子系统进行适当的访问，否则配置可能会失败。

```
allow (read) user="anybody";allow (modify) group="Subsystem Group"
```

表 D.42. certServer.securitydomain.domainxml ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看安全域配置。	Allow	任何人
modify	通过更改实例信息以及添加和删除实例来修改安全域配置。	Allow	子系统组 企业管理员

D.4. 特定于密钥恢复机构的 ACL

本节介绍了特定于 KRA 的默认访问控制配置。KRA ACL 配置还包括 第 D.2 节 “常见 ACL” 中列出的所有常见 ACL。

每个 KRA 接口（管理控制台和代理及端点服务页面）以及列出和下载密钥等常见操作都有访问控制规

则。

D.4.1. certServer.job.configuration

控制谁可以为 KRA 配置作业。

> *allow (read) group="Administrators" || group="Key Recovery Authority Agents" || group="Auditors";allow (modify) group="Administrators"*

表 D.43. certServer.job.configuration ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看基本作业设置、作业实例设置和作业插件设置。列出作业插件和作业实例。	Allow	管理员 代理 审核员
modify	添加和删除作业插件和作业实例。修改作业插件和作业实例。	Allow	管理员

D.4.2. certServer.kra.certificate.transport

控制谁可以查看 KRA 的传输证书。

> *allow (read) user="anybody"*

表 D.44. certServer.kra.certificate.transport ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看 KRA 实例的传输证书。	Allow	任何人

D.4.3. certServer.kra.configuration

控制谁可以为 KRA 配置和管理设置。

> *allow (read) group="Administrators" || group="Auditors" || group="Key Recovery Authority Agents" || allow (modify) group="Administrators"*

表 D.45. certServer.kra.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	阅读所需的恢复代理批准数量。	Allow	管理员 代理 审核员
modify	更改所需的恢复代理批准数量。	Allow	管理员

D.4.4. certServer.kra.connector

控制哪些实体可以通过 CA 上配置的特殊连接器提交请求以连接到 KRA。默认配置是：

```
allow (submit) group="Trusted Managers"
```

表 D.46. certServer.kra.connector ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	提交新的密钥归档请求（仅用于非短信）。	Allow	可信管理器

D.4.5. certServer.kra.GenerateKeyPair

控制谁可以对 KRA 提交密钥恢复请求。默认配置是：

```
allow (execute) group="Key Recovery Authority Agents"
```

表 D.47. certServer.kra.GenerateKeyPair ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	执行服务器端密钥生成（仅限TMS）。	Allow	KRA Agents

D.4.6. certServer.kra.getTransportCert

控制谁可以对 KRA 提交密钥恢复请求。默认配置是：

> *allow (download) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" || group="Enterprise TPS Administrators"*

表 D.48. certServer.kra.getTransportCert ACL Summary

操作	描述	allow/Deny Access	目标用户/组
下载	检索 KRA 传输证书。	Allow	企业管理员

D.4.7. certServer.kra.group

控制对为 KRA 实例添加用户和组的内部数据库的访问。

> *allow (modify,read) group="Administrators"*

表 D.49. certServer.kra.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	为实例创建、编辑或删除用户和组条目。	Allow	管理员
读取	查看实例的用户和组条目。	Allow	管理员

D.4.8. certServer.kra.key

通过查看、恢复或下载密钥控制谁可以访问密钥信息。默认配置是：

> *allow (read,recover,download) group="Key Recovery Authority Agents"*

表 D.50. certServer.kra.key ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	显示有关密钥存档记录的公共信息。	Allow	KRA Agents
recover	从数据库检索关键信息以执行恢复操作。	Allow	KRA Agents

操作	描述	allow/Deny Access	目标用户/组
下载	通过代理服务页面下载密钥信息。	Allow	KRA Agents

D.4.9. certServer.kra.keys

控制可以通过代理服务页面列出归档的密钥。

> *allow (list) group="Key Recovery Authority Agents"*

表 D.51. certServer.kra.keys ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	搜索和列出归档密钥的范围。	Allow	KRA Agents

D.4.10. certServer.kra.registerUser

定义哪个组或用户可为实例创建代理用户。默认配置是：

> *allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" || group="Enterprise TPS Administrators"*

表 D.52. certServer.kra.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	注册新用户。	Allow	企业管理员
读取	读取现有用户信息.	Allow	企业管理员

D.4.11. certServer.kra.request

控制谁可以在代理服务界面中查看密钥归档和恢复请求。

> *allow (read) group="Key Recovery Authority Agents"*

表 D.53. certServer.kra.request ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看密钥存档或恢复请求。	Allow	KRA Agents

D.4.12. certServer.kra.request.status

控制谁可以在端点页面中查看密钥恢复请求的状态。

> *allow (read) group="Key Recovery Authority Agents"*

表 D.54. certServer.kra.request.status ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	在代理服务页面中检索密钥恢复请求的状态。	Allow	KRA Agents

D.4.13. certServer.kra.requests

控制在代理服务界面中列出密钥归档和恢复请求的人员。

> *allow (list) group="Key Recovery Authority Agents"*

表 D.55. certServer.kra.requests ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	检索关键存档和恢复请求范围的详细信息。	Allow	KRA Agents

D.4.14. certServer.kra.systemstatus

控制谁可以查看 KRA 实例的统计信息。

> *allow (read) group="Key Recovery Authority Agents"*

表 D.56. certServer.kra.systemstatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看统计数据。	Allow	KRA Agents

D.4.15. certServer.kra.TokenKeyRecovery

控制谁可以将令牌的密钥恢复请求提交到 KRA。这是替换丢失令牌的常见请求。默认配置是：

```
allow (submit) group="Key Recovery Authority Agents"
```

表 D.57. certServer.kra.TokenKeyRecovery ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	提交或初始化令牌恢复请求。	Allow	KRA Agents

D.5. 在线证书状态管理器特定 ACL

本节涵盖为在线证书 Status Manager 设置的默认访问控制配置属性。OCSP 响应者的 ACL 配置还包括 第 D.2 节 “常见 ACL” 中列出的所有常见 ACL。

每个 OCSP 的接口（管理控制台和代理及端点服务页面）以及列出和下载 CRL 等常见操作都有访问控制规则。

D.5.1. certServer.ee.crl

通过最终用户页面控制对 CRL 的访问。

```
allow (read) user="anybody"
```

表 D.58. certServer.ee.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	检索并查看证书撤销列表。	Allow	任何人

D.5.2. certServer.ee.request.ocsp

根据客户端提交 OCSP 请求的 IP 地址控制访问。

```
allow (submit) ipaddress=".*"
```

表 D.59. certServer.ee.request.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
submit	提交 OCSP 请求。	Allow	所有 IP 地址

D.5.3. certServer.ocsp.ca

控制可以指示 OCSP 响应者的人员。默认设置为：

> *allow (add) group="Online Certificate Status Manager Agents"*

表 D.60. certServer.ocsp.ca ACL Summary

操作	描述	allow/Deny Access	目标用户/组
添加	指示 OCSP 响应者对新 CA 的 OCSP 请求做出响应。	Allow	OCSP Manager 代理

D.5.4. certServer.ocsp.cas

控制可在代理服务界面中列出谁，将 CRL 发布到在线证书 Status Manager 的所有证书管理器。默认设置为：

> *allow (list) group="Online Certificate Status Manager Agents"*

表 D.61. certServer.ocsp.cas ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	列出将 CRL 发布到 OCSP 响应程序的所有证书管理器。	Allow	代理

D.5.5. certServer.ocsp.certificate

控制谁能够验证证书的状态。默认设置为：

> *allow (validate) group="Online Certificate Status Manager Agents"*

表 D.62. certServer.ocsp.certificate ACL Summary

操作	描述	allow/Deny Access	目标用户/组
validate	验证指定证书的状态。	Allow	OCSP 代理

D.5.6. certServer.ocsp.configuration

控制谁可以访问、查看或修改证书管理器的 OCSP 服务的配置。默认配置是：

> *allow (read) group="Administrators" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"*

表 D.63. certServer.ocsp.configuration ACL 摘要

操作	描述	allow/Deny Access	目标用户/组
读取	查看 OCSP 插件信息、OCSP 配置和 OCSP 存储配置。列出 OCSP 存储配置。	Allow	管理员 在线证书状态管理器代理 审核员
modify	修改 OCSP 配置，OCSP 存储配置以及默认的 OCSP 存储。	Allow	管理员

D.5.7. certServer.ocsp.crl

通过代理服务接口控制读取或更新 CRL 的访问。默认设置为：

> *allow (add) group="Online Certificate Status Manager Agents" || group="Trusted Managers"*

表 D.64. certServer.ocsp.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
add	向由 OCSP 响应程序管理的用户添加新的 CRL。	Allow	OCSP 代理 可信管理器

D.5.8. certServer.ocsp.group

控制内部数据库的访问权限，以便为在线证书 Status Manager 实例添加用户和组。

```
allow (modify,read) group="Administrators"
```

表 D.65. certServer.ocsp.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	创建、编辑或删除实例的用户和组条目。	Allow	管理员
读取	查看实例的用户和组条目。	Allow	管理员

D.5.9. certServer.ocsp.info

控制谁可以读取 OCSP 响应者的信息。

```
allow (read) group="Online Certificate Status Manager Agents"
```

表 D.66. certServer.ocsp.info ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看 OCSP 响应器信息。	Allow	OCSP 代理

D.6. 基于令牌密钥服务的 ACL

本节涵盖为 Token Key Service(TKS)设置的默认访问控制配置属性。TKS ACL 配置还包括 第 D.2 节 “常见 ACL” 中列出的所有常见 ACL。

为 TKS 的管理控制台设置了访问控制规则，以及由其他子系统访问 TKS 的访问。

D.6.1. certServer.tks.encrypteddata

控制谁可以加密数据。

```
allow(execute) group="Token Key Service Manager Agents"
```

表 D.67. certServer.tks.encrypteddata ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	加密数据存储在 TKS 中。	Allow	TKS 代理

D.6.2. certServer.tks.group

控制对为 TKS 实例添加用户和组的内部数据库的访问。

> *allow (modify,read) group="Administrators"*

表 D.68. certServer.tks.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	为实例创建、编辑或删除用户和组条目。	Allow	管理员
读取	查看实例的用户和组条目。	Allow	管理员

D.6.3. certServer.tks.importTransportCert

控制谁可以导入由 TKS 用来提供密钥的传输证书。

> *allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" || group="Enterprise TPS Administrators"*

表 D.69. certServer.tks.importTransportCert ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	更新传输证书。	Allow	企业管理员
读取	导入传输证书。	Allow	企业管理员

D.6.4. certServer.tks.keysetdata

控制者可以查看由 TKS 派生和存储的有关密钥集的信息。

> *allow (execute) group="Token Key Service Manager Agents"*

表 D.70. certServer.tks.keysetdata ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	创建分散的密钥集合数据。	Allow	TKS 代理

D.6.5. certServer.tks.registerUser

定义哪个组或用户可为实例创建代理用户。默认配置是：

> *allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" || group="Enterprise TPS Administrators"*

表 D.71. certServer.tks.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
modify	注册新代理。	Allow	企业管理员
读取	读取现有代理信息。	Allow	企业管理员

D.6.6. certServer.tks.sessionkey

控制谁可以创建连接到 TPS 实例使用的会话密钥。

> *allow (execute) group="Token Key Service Manager Agents"*

表 D.72. certServer.tks.sessionkey ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	创建由 TKS 生成的会话密钥。	Allow	TKS 代理

D.6.7. certServer.tks.randomdata

控制谁可以创建随机数据。

> *allow (execute) group="Token Key Service Manager Agents"*

-

表 D.73. certServer.tks.randomdata ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	生成随机数据。	Allow	TKS 代理

附录 E. 审计事件

本附录提供了单个审计事件及其参数描述和格式。日志中的每个审计事件都带有以下信息：

- 线程的 Java 标识符。例如：

  > 0.localhost-startStop-1

- 事件发生的时间戳。例如：

  > [21/Jan/2019:17:53:00 IST]

- 日志源（14 是 SIGNED_AUDIT）：

  > [14]

- 当前日志级别（6 是与安全相关的事件）。请参阅 红帽认证系统 9 规划、安装和部署指南中的 日志级别(Message Categories) 部分。例如：

  > [6]

- 日志事件（特定于日志事件）的信息；有关特定日志事件中每个字段的信息，请参阅 第 E.1 节 “审计事件描述”。例如：

  > [AuditEvent=AUDIT_LOG_STARTUP][SubjectID=$System$][Outcome=Success] audit function startup

E.1. 审计事件描述

下表列出了证书系统中提供的审计事件：

```
########################## SIGNED AUDIT EVENTS ###############################
# Common fields:
# - Outcome: "Success" or "Failure"
# - SubjectID: The UID of the user responsible for the operation
#     "$System$" or "SYSTEM" if system-initiated operation (e.g. log signing).
#
```

```
    ##############################################################################
    # Required Audit Events
    #
    # Event: ACCESS_SESSION_ESTABLISH with [Outcome=Failure]
    # Description: This event is used when access session failed to establish.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientIP: Client IP address.
    # - ServerIP: Server IP address.
    # - SubjectID: Client certificate subject DN.
    # - Outcome: Failure
    # - Info: Failure reason.
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_ESTABLISH_FAILURE=\
    <type=ACCESS_SESSION_ESTABLISH>:[AuditEvent=ACCESS_SESSION_ESTABLISH]{0}
access session establish failure
    #
    # Event: ACCESS_SESSION_ESTABLISH with [Outcome=Success]
    # Description: This event is used when access session was established successfully.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientIP: Client IP address.
    # - ServerIP: Server IP address.
    # - SubjectID: Client certificate subject DN.
    # - Outcome: Success
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_ESTABLISH_SUCCESS=\
    <type=ACCESS_SESSION_ESTABLISH>:[AuditEvent=ACCESS_SESSION_ESTABLISH]{0}
access session establish success
    #
    # Event: ACCESS_SESSION_TERMINATED
    # Description: This event is used when access session was terminated.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientIP: Client IP address.
    # - ServerIP: Server IP address.
    # - SubjectID: Client certificate subject DN.
    # - Info: The TLS Alert received from NSS
    # - Outcome: Success
    # - Info: The TLS Alert received from NSS
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_TERMINATED=\
    <type=ACCESS_SESSION_TERMINATED>:[AuditEvent=ACCESS_SESSION_TERMINATED]
{0} access session terminated
    #
    # Event: AUDIT_LOG_SIGNING
    # Description: This event is used when a signature on the audit log is generated (same as
"flush" time).
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: Predefined to be "$System$" because this operation
    #     associates with no user.
```

```
    # - Outcome: Success
    # - sig: The base-64 encoded signature of the buffer just flushed.
    #
    LOGGING_SIGNED_AUDIT_AUDIT_LOG_SIGNING_3=[AuditEvent=AUDIT_LOG_SIGNING]
[SubjectID={0}][Outcome={1}] signature of audit buffer just flushed: sig: {2}
    #
    # Event: AUDIT_LOG_STARTUP
    # Description: This event is used at audit function startup.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    #
    LOGGING_SIGNED_AUDIT_AUDIT_LOG_STARTUP_2=<type=AUDIT_LOG_STARTUP>:
[AuditEvent=AUDIT_LOG_STARTUP][SubjectID={0}][Outcome={1}] audit function startup
    #
    # Event: AUTH with [Outcome=Failure]
    # Description: This event is used when authentication fails.
    #   In case of TLS-client auth, only webserver env can pick up the TLS violation.
    #   CS authMgr can pick up certificate mismatch, so this event is used.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID:
    # - Outcome: Failure
    #     (obviously, if authentication failed, you won't have a valid SubjectID, so
    #     in this case, SubjectID should be $Unidentified$)
    # - AuthMgr: The authentication manager instance name that did
    #     this authentication.
    # - AttemptedCred: The credential attempted and failed.
    #
    LOGGING_SIGNED_AUDIT_AUTH_FAIL=<type=AUTH>:[AuditEvent=AUTH]{0}
authentication failure
    #
    # Event: AUTH with [Outcome=Success]
    # Description: This event is used when authentication succeeded.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of user who has been authenticated
    # - Outcome: Success
    # - AuthMgr: The authentication manager instance name that did
    #     this authentication.
    #
    LOGGING_SIGNED_AUDIT_AUTH_SUCCESS=<type=AUTH>:[AuditEvent=AUTH]{0}
authentication success
    #
    # Event: AUTHZ with [Outcome=Failure]
    # Description: This event is used when authorization has failed.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of user who has failed to be authorized for an action
    # - Outcome: Failure
    # - aclResource: The ACL resource ID as defined in ACL resource list.
```

```
    # - Op: One of the operations as defined with the ACL statement
    #    e.g. "read" for an ACL statement containing "(read,write)".
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_AUTHZ_FAIL=<type=AUTHZ>:[AuditEvent=AUTHZ]{0}
authorization failure
    #
    # Event: AUTHZ with [Outcome=Success]
    # Description: This event is used when authorization is successful.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of user who has been authorized for an action
    # - Outcome: Success
    # - aclResource: The ACL resource ID as defined in ACL resource list.
    # - Op: One of the operations as defined with the ACL statement
    #     e.g. "read" for an ACL statement containing "(read,write)".
    #
    LOGGING_SIGNED_AUDIT_AUTHZ_SUCCESS=<type=AUTHZ>:[AuditEvent=AUTHZ]{0}
authorization success
    #
    # Event: CERT_PROFILE_APPROVAL
    # Description: This event is used when an agent approves/disapproves a certificate profile
set by the
    #   administrator for automatic approval.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of the CA agent who approved the certificate enrollment profile
    # - Outcome:
    # - ProfileID: One of the profiles defined by the administrator
    #     and to be approved by an agent.
    # - Op: "approve" or "disapprove".
    #
    LOGGING_SIGNED_AUDIT_CERT_PROFILE_APPROVAL_4=
<type=CERT_PROFILE_APPROVAL>:[AuditEvent=CERT_PROFILE_APPROVAL][SubjectID=
{0}][Outcome={1}][ProfileID={2}][Op={3}] certificate profile approval
    #
    # Event: CERT_REQUEST_PROCESSED
    # Description: This event is used when certificate request has just been through the
approval process.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of the agent who approves, rejects, or cancels
    #     the certificate request.
    # - Outcome:
    # - ReqID: The request ID.
    # - InfoName: "certificate" (in case of approval), "rejectReason"
    #     (in case of reject), or "cancelReason" (in case of cancel)
    # - InfoValue: The certificate (in case of success), a reject reason in
    #     text, or a cancel reason in text.
    # - CertSerialNum:
    #
    LOGGING_SIGNED_AUDIT_CERT_REQUEST_PROCESSED=
<type=CERT_REQUEST_PROCESSED>:[AuditEvent=CERT_REQUEST_PROCESSED]{0}
```

```
certificate request processed
   #
   # Event: CERT_SIGNING_INFO
   # Description: This event indicates which key is used to sign certificates.
   # Applicable subsystems: CA
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: $System$
   # - Outcome: Success
   # - SKI: Subject Key Identifier of the certificate signing certificate
   # - AuthorityID: (applicable only to lightweight CA)
   #
   LOGGING_SIGNED_AUDIT_CERT_SIGNING_INFO=<type=CERT_SIGNING_INFO>:
[AuditEvent=CERT_SIGNING_INFO]{0} certificate signing info
   #
   # Event: CERT_STATUS_CHANGE_REQUEST
   # Description: This event is used when a certificate status change request (e.g. revocation)
   #   is made (before approval process).
   # Applicable subsystems: CA
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of uer who performed the action
   # - Outcome:
   # - ReqID: The request ID.
   # - CertSerialNum: The serial number (in hex) of the certificate to be revoked.
   # - RequestType: "revoke", "on-hold", "off-hold"
   #
   LOGGING_SIGNED_AUDIT_CERT_STATUS_CHANGE_REQUEST=
<type=CERT_STATUS_CHANGE_REQUEST>:
[AuditEvent=CERT_STATUS_CHANGE_REQUEST]{0} certificate revocation/unrevocation
request made
   #
   # Event: CERT_STATUS_CHANGE_REQUEST_PROCESSED
   # Description: This event is used when certificate status is changed (revoked, expired, on-
hold,
   #   off-hold).
   # Applicable subsystems: CA
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: The UID of the agent that processed the request.
   # - Outcome:
   # - ReqID: The request ID.
   # - RequestType: "revoke", "on-hold", "off-hold"
   # - Approval: "complete", "rejected", or "canceled"
   #     (note that "complete" means "approved")
   # - CertSerialNum: The serial number (in hex).
   # - RevokeReasonNum: One of the following number:
   #     reason number       reason
   #     ------------------------------------
   #     0             Unspecified
   #     1             Key compromised
   #     2             CA key compromised (should not be used)
   #     3             Affiliation changed
   #     4             Certificate superceded
   #     5             Cessation of operation
   #     6             Certificate is on-hold
```

```
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_CERT_STATUS_CHANGE_REQUEST_PROCESSED=
<type=CERT_STATUS_CHANGE_REQUEST_PROCESSED>:
[AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED]{0} certificate status change
request processed
    #
    # Event: CLIENT_ACCESS_SESSION_ESTABLISH with [Outcome=Failure]
    # Description: This event is when access session failed to establish when Certificate
System acts as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Failure
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_ESTABLISH_FAILURE=\
    <type=CLIENT_ACCESS_SESSION_ESTABLISH>:
[AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH]{0} access session failed to establish
when Certificate System acts as client
    #
    # Event: CLIENT_ACCESS_SESSION_ESTABLISH with [Outcome=Success]
    # Description: This event is used when access session was established successfully when
    #   Certificate System acts as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Success
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_ESTABLISH_SUCCESS=\
    <type=CLIENT_ACCESS_SESSION_ESTABLISH>:
[AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH]{0} access session establish
successfully when Certificate System acts as client
    #
    # Event: CLIENT_ACCESS_SESSION_TERMINATED
    # Description: This event is used when access session was terminated when Certificate
System acts as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Success
    # - Info: The TLS Alert received from NSS
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_TERMINATED=\
```

```
    <type=CLIENT_ACCESS_SESSION_TERMINATED>:
[AuditEvent=CLIENT_ACCESS_SESSION_TERMINATED]{0} access session terminated when
Certificate System acts as client
    #
    # Event: CMC_REQUEST_RECEIVED
    # Description: This event is used when a CMC request is received.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    #     If CMC requests is signed by an agent, SubjectID should
    #     be that of the agent.
    #     In case of an unsigned request, it would bear $Unidentified$.
    # - Outcome:
    # - CMCRequest: Base64 encoding of the CMC request received
    #
    LOGGING_SIGNED_AUDIT_CMC_REQUEST_RECEIVED_3=
<type=CMC_REQUEST_RECEIVED>:[AuditEvent=CMC_REQUEST_RECEIVED][SubjectID={0}]
[Outcome={1}][CMCRequest={2}] CMC request received
    #
    # Event: CMC_RESPONSE_SENT
    # Description: This event is used when a CMC response is sent.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    # - Outcome:
    # - CMCResponse: Base64 encoding of the CMC response sent
    #
    LOGGING_SIGNED_AUDIT_CMC_RESPONSE_SENT_3=<type=CMC_RESPONSE_SENT>:
[AuditEvent=CMC_RESPONSE_SENT][SubjectID={0}][Outcome={1}][CMCResponse={2}] CMC
response sent
    #
    # Event: CMC_SIGNED_REQUEST_SIG_VERIFY
    # Description: This event is used when agent signed CMC certificate requests or revocation
requests
    #   are submitted and signature is verified.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: the user who signed the CMC request (success case)
    # - Outcome:
    # - ReqType: The request type (enrollment, or revocation).
    # - CertSubject: The certificate subject name of the certificate request.
    # - SignerInfo: A unique String representation for the signer.
    #
    LOGGING_SIGNED_AUDIT_CMC_SIGNED_REQUEST_SIG_VERIFY=
<type=CMC_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_SIGNED_REQUEST_SIG_VERIFY]{0} agent signed CMC request signature
verification
    #
    # Event: CMC_USER_SIGNED_REQUEST_SIG_VERIFY
    # Description: This event is used when CMC (user-signed or self-signed) certificate
requests or revocation requests
    #   are submitted and signature is verified.
    # Applicable subsystems: CA
```

```
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: the user who signed the CMC request (success case)
    # - Outcome:
    # - ReqType: The request type (enrollment, or revocation).
    # - CertSubject: The certificate subject name of the certificate request.
    # - CMCSignerInfo: A unique String representation for the CMC request signer.
    # - info:
    #
    LOGGING_SIGNED_AUDIT_CMC_USER_SIGNED_REQUEST_SIG_VERIFY_FAILURE=
<type=CMC_USER_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_USER_SIGNED_REQUEST_SIG_VERIFY]{0} User signed CMC request
signature verification failure
    LOGGING_SIGNED_AUDIT_CMC_USER_SIGNED_REQUEST_SIG_VERIFY_SUCCESS=
<type=CMC_USER_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_USER_SIGNED_REQUEST_SIG_VERIFY]{0} User signed CMC request
signature verification success
    #
    # Event: CONFIG_ACL
    # Description: This event is used when configuring ACL information.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_ACL_3=<type=CONFIG_ACL>:
[AuditEvent=CONFIG_ACL][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] ACL
configuration parameter(s) change
    #
    # Event: CONFIG_AUTH
    # Description: This event is used when configuring authentication.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #     --- Password MUST NOT be logged ---
    #
    LOGGING_SIGNED_AUDIT_CONFIG_AUTH_3=<type=CONFIG_AUTH>:
[AuditEvent=CONFIG_AUTH][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}]
authentication configuration parameter(s) change
    #
    # Event: CONFIG_CERT_PROFILE
    # Description: This event is used when configuring certificate profile
    #   (general settings and certificate profile).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
```

```
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_CERT_PROFILE_3=<type=CONFIG_CERT_PROFILE>:
[AuditEvent=CONFIG_CERT_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs=
{2}] certificate profile configuration parameter(s) change
    #
    # Event: CONFIG_CRL_PROFILE
    # Description: This event is used when configuring CRL profile
    #   (extensions, frequency, CRL format).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_CRL_PROFILE_3=<type=CONFIG_CRL_PROFILE>:
[AuditEvent=CONFIG_CRL_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}]
CRL profile configuration parameter(s) change
    #
    # Event: CONFIG_DRM
    # Description: This event is used when configuring KRA.
    #   This includes key recovery scheme, change of any secret component.
    # Applicable subsystems: KRA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #     --- secret component (password) MUST NOT be logged ---
    #
    LOGGING_SIGNED_AUDIT_CONFIG_DRM_3=<type=CONFIG_DRM>:
[AuditEvent=CONFIG_DRM][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] DRM
configuration parameter(s) change
    #
    # Event: CONFIG_OCSP_PROFILE
    # Description: This event is used when configuring OCSP profile
    #   (everything under Online Certificate Status Manager).
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_OCSP_PROFILE_3=<type=CONFIG_OCSP_PROFILE>:
[AuditEvent=CONFIG_OCSP_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs=
```

```
{2}] OCSP profile configuration parameter(s) change
   #
   # Event: CONFIG_ROLE
   # Description: This event is used when configuring role information.
   #   This includes anything under users/groups, add/remove/edit a role, etc.
   # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #
   LOGGING_SIGNED_AUDIT_CONFIG_ROLE=<type=CONFIG_ROLE>:
[AuditEvent=CONFIG_ROLE]{0} role configuration parameter(s) change
   #
   # Event: CONFIG_SERIAL_NUMBER
   # Description: This event is used when configuring serial number ranges
   #   (when requesting a serial number range when cloning, for example).
   # Applicable subsystems: CA, KRA
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #
   LOGGING_SIGNED_AUDIT_CONFIG_SERIAL_NUMBER_1=
<type=CONFIG_SERIAL_NUMBER>:[AuditEvent=CONFIG_SERIAL_NUMBER][SubjectID={0}]
[Outcome={1}][ParamNameValPairs={2}] serial number range update
   #
   # Event: CONFIG_SIGNED_AUDIT
   # Description: This event is used when configuring signedAudit.
   # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #
   LOGGING_SIGNED_AUDIT_CONFIG_SIGNED_AUDIT=<type=CONFIG_SIGNED_AUDIT>:
[AuditEvent=CONFIG_SIGNED_AUDIT]{0} signed audit configuration parameter(s) change
   #
   # Event: CONFIG_TRUSTED_PUBLIC_KEY
   # Description: This event is used when:
   #   1. "Manage Certificate" is used to edit the trustness of certificates
   #      and deletion of certificates
   #   2. "Certificate Setup Wizard" is used to import CA certificates into the
   #      certificate database (Although CrossCertificatePairs are stored
   #      within internaldb, audit them as well)
   # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
   # Enabled by default: Yes
```

```
    # Fields:
    # - SubjectID: ID of administrator who performed this configuration
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_TRUSTED_PUBLIC_KEY=
<type=CONFIG_TRUSTED_PUBLIC_KEY>:[AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY]{0}
certificate database configuration
    #
    # Event: CRL_SIGNING_INFO
    # Description: This event indicates which key is used to sign CRLs.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    # - SKI: Subject Key Identifier of the CRL signing certificate
    #
    LOGGING_SIGNED_AUDIT_CRL_SIGNING_INFO=<type=CRL_SIGNING_INFO>:
[AuditEvent=CRL_SIGNING_INFO]{0} CRL signing info
    #
    # Event: DELTA_CRL_GENERATION
    # Description: This event is used when delta CRL generation is complete.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $Unidentified$
    # - Outcome: "Success" when delta CRL is generated successfully, "Failure" otherwise.
    # - CRLnum: The CRL number that identifies the CRL
    # - Info:
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_DELTA_CRL_GENERATION=
<type=DELTA_CRL_GENERATION>:[AuditEvent=DELTA_CRL_GENERATION]{0} Delta CRL
generation
    #
    # Event: FULL_CRL_GENERATION
    # Description: This event is used when full CRL generation is complete.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome: "Success" when full CRL is generated successfully, "Failure" otherwise.
    # - CRLnum: The CRL number that identifies the CRL
    # - Info:
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_FULL_CRL_GENERATION=<type=FULL_CRL_GENERATION>:
[AuditEvent=FULL_CRL_GENERATION]{0} Full CRL generation
    #
    # Event: PROFILE_CERT_REQUEST
    # Description: This event is used when a profile certificate request is made (before approval
process).
    # Applicable subsystems: CA
```

```
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    #     If CMC enrollment requests signed by an agent, SubjectID should
    #     be that of the agent.
    # - Outcome:
    # - CertSubject: The certificate subject name of the certificate request.
    # - ReqID: The certificate request ID.
    # - ProfileID: One of the certificate profiles defined by the
    #     administrator.
    #
    LOGGING_SIGNED_AUDIT_PROFILE_CERT_REQUEST_5=
<type=PROFILE_CERT_REQUEST>:[AuditEvent=PROFILE_CERT_REQUEST][SubjectID={0}]
[Outcome={1}][ReqID={2}][ProfileID={3}][CertSubject={4}] certificate request made with
certificate profiles
    #
    # Event: PROOF_OF_POSSESSION
    # Description: This event is used for proof of possession during certificate enrollment
processing.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id that represents the authenticated user
    # - Outcome:
    # - Info: some information on when/how it occurred
    #
    LOGGING_SIGNED_AUDIT_PROOF_OF_POSSESSION_3=
<type=PROOF_OF_POSSESSION>:[AuditEvent=PROOF_OF_POSSESSION][SubjectID={0}]
[Outcome={1}][Info={2}] proof of possession
    #
    # Event: OCSP_ADD_CA_REQUEST_PROCESSED
    # Description: This event is used when an add CA request to the OCSP Responder is
processed.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: "Success" when CA is added successfully, "Failure" otherwise.
    # - CASubjectDN: The subject DN of the leaf CA cert in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_ADD_CA_REQUEST_PROCESSED=
<type=OCSP_ADD_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_ADD_CA_REQUEST_PROCESSED]{0} Add CA for OCSP Responder
    #
    # Event: OCSP_GENERATION
    # Description: This event is used when an OCSP response generated is complete.
    # Applicable subsystems: CA, OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $NonRoleUser$
    # - Outcome: "Success" when OCSP response is generated successfully, "Failure"
otherwise.
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_OCSP_GENERATION=<type=OCSP_GENERATION>:
[AuditEvent=OCSP_GENERATION]{0} OCSP response generation
```

```
    #
    # Event: OCSP_REMOVE_CA_REQUEST_PROCESSED with [Outcome=Failure]
    # Description: This event is used when a remove CA request to the OCSP Responder is
processed and failed.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: Failure
    # - CASubjectDN: The subject DN of the leaf CA certificate in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_REMOVE_CA_REQUEST_PROCESSED_FAILURE=
<type=OCSP_REMOVE_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_REMOVE_CA_REQUEST_PROCESSED]{0} Remove CA for OCSP
Responder has failed
    #
    # Event: OCSP_REMOVE_CA_REQUEST_PROCESSED with [Outcome=Success]
    # Description: This event is used when a remove CA request to the OCSP Responder is
processed successfully.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: "Success" when CA is removed successfully, "Failure" otherwise.
    # - CASubjectDN: The subject DN of the leaf CA certificate in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_REMOVE_CA_REQUEST_PROCESSED_SUCCESS=
<type=OCSP_REMOVE_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_REMOVE_CA_REQUEST_PROCESSED]{0} Remove CA for OCSP
Responder is successful
    #
    # Event: OCSP_SIGNING_INFO
    # Description: This event indicates which key is used to sign OCSP responses.
    # Applicable subsystems: CA, OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    # - SKI: Subject Key Identifier of the OCSP signing certificate
    # - AuthorityID: (applicable only to lightweight CA)
    #
    LOGGING_SIGNED_AUDIT_OCSP_SIGNING_INFO=<type=OCSP_SIGNING_INFO>:
[AuditEvent=OCSP_SIGNING_INFO]{0} OCSP signing info
    #
    # Event: ROLE_ASSUME
    # Description: This event is used when a user assumes a role.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID:
    # - Outcome:
    # - Role: One of the valid roles:
    #     "Administrators", "Certificate Manager Agents", or "Auditors".
    #     Note that customized role names can be used once configured.
    #
    LOGGING_SIGNED_AUDIT_ROLE_ASSUME=<type=ROLE_ASSUME>:
```

```
[AuditEvent=ROLE_ASSUME]{0} assume privileged role
    #
    # Event: SECURITY_DOMAIN_UPDATE
    # Description: This event is used when updating contents of security domain
    #   (add/remove a subsystem).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: CA administrator user ID
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_SECURITY_DOMAIN_UPDATE_1=
<type=SECURITY_DOMAIN_UPDATE>:[AuditEvent=SECURITY_DOMAIN_UPDATE][SubjectID=
{0}][Outcome={1}][ParamNameValPairs={2}] security domain update
    #
    # Event: SELFTESTS_EXECUTION
    # Description: This event is used when self tests are run.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    #
    LOGGING_SIGNED_AUDIT_SELFTESTS_EXECUTION_2=<type=SELFTESTS_EXECUTION>:
[AuditEvent=SELFTESTS_EXECUTION][SubjectID={0}][Outcome={1}] self tests execution (see
selftests.log for details)
```

术语表

A

administrator

安装并配置一个或多个证书证书证书的人员；系统管理器并为其设置特权用户或代理。另请参阅 agent。

agent

属于一组授权来管理证书证书证书系统nbsp 的组的用户；System manager. 代理服务另请参阅证书管理器代理、密钥恢复授权代理。

APDU

应用程序协议数据单元。一个通信单元（类似于字节），用于智能卡和智能卡读取器之间的通信。

auditor

此特权用户，可以查看签名的审计日志。

代理批准的注册

在发布证书前，需要代理批准请求的注册。

代理服务

1. 可通过证书证书系统nbsp 管理的服务；系统 agent 通过证书证书系统nbsp 提供的 HTML 页面；为代理分配了必要的权限的系统子系统。

2.用于管理此类服务的 HTML 页面。

审计日志

记录各种系统事件的日志。此日志可以签名，提供不被篡改的证明，并且只能被审核员用户读取。

属性值断言(AVA)

表单 属性 = value 的断言，其中 属性是 标签，如 o (organization)或 uid (user ID)，值是 "Red Hatnbsp;Hat, Inc." 或登录名称等的值。AVAs 用于形成标识证书的主体的 可分辨名称(DN)，称为证书的 主题名称。

授权

访问由服务器控制的资源的权限。授权通常会在服务器评估与资源关联的 ACL 后进行。请参阅 访问控制列表(ACL)。

自动注册

配置证书证书系统空间的方法；System 子系统允许自动进行终端注册，而无需人为干预。通过这种身份验证，成功完成身份验证模块的处理的证书请求将自动批准以进行配置集处理，并且证书能够保证。

访问控制

控制允许特定用户执行的操作的进程。例如，对服务器的访问控制通常基于一个由密码或证书建立的身份，以及有关该实体可以执行的操作的规则。另请参阅 访问控制列表(ACL)。

访问控制列表(ACL)

访问控制条目的集合，该条目定义服务器收到对特定资源的请求时要评估的访问层次结构。请参阅 访问控制指令(ACI)。

访问控制指令(ACI)

访问规则，指定如何识别请求访问权限的主题或特定主题允许或拒绝什么权限。请参阅 访问控制列表(ACL)。

身份验证

信心识别；确保某些计算机化交易的方不是成名的交易。身份验证通常涉及使用密码、证书、PIN 或其他信息来验证计算机网络上的身份。另请参阅 基于密码的身份验证、基于证书的验证、客户端身份验证、服务器身份验证。

身份验证模块

一组规则（实施为 Java™ 类），用于对最终实体、代理、管理员或其他需要与之交互的实体进行身份验证；系统子系统。对于典型的最终用户注册，在用户提供了注册表请求的信息后，注册 servlet 使用与该表单关联的身份验证模块来验证信息并验证用户身份。请参阅 servlet。

高级加密标准(AES)

高级加密标准(AES)与其前身数据加密标准(DES)是 FIPS 批准的对称加密标准。AES 由 2002 年美国政府采用。它定义三个块密码： AES-128、AES-192 和 AES-256。美国国家标准与技术研究所(NIST)在美国中定义了 AES 标准。FIPS PUB 197。如需更多信息，请参阅 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf。

B

bind DN

用户 ID，格式为可分辨名称(DN)，密码用于向 Red Hat Directory Server 进行身份验证。

C

CA 层次结构

root CA 将授权签发证书到下级 CA 的 CA 的层次结构。从属 CA 也可通过委派发出状态到其他 CA 来扩展层次结构。另请参阅 证书颁发机构(CA)、subordinate CA、Root CA。

CA 服务器密钥

提供 CA 服务的服务器的 SSL 服务器密钥。

CA 签名密钥

与 CA 证书中的公钥对应的私钥。CA 使用它的签名密钥来签署证书和 CRL。

CA 证书

标识证书颁发机构的证书。另请参阅 证书颁发机构(CA)、subordinate CA、Root CA。

certificate

根据 X.509 标准格式的数字数据，指定独立、公司或其他实体的名称（证书 主题名称 ）并验证 public key 也包含在该实体（也属于该实体）。证书由 证书颁发机构(CA) 发布并用数字签名。可以通过通过 数字签名 技术检查 CA 的 public-key cryptography 来验证证书的有效性。要在 公钥基础架构(PKI) 中信任，该证书必须由在 PKI 中注册的其他实体信任的 CA 发布并签名。

Certificate System

请参阅 RedRed Hat Certificate Systemnbsp;Hat CertificateRed Hat Certificate Systemnbsp;System、加密消息语法(CS)。

cipher

请参阅 cryptographic algorithm。

CMC

请参阅 通过 Cryptographic 消息语法(CMC)的证书管理消息。

CMC 注册

允许使用代理的签名证书将签名注册或经过签名的撤销请求发送到证书管理器的功能。这些请求随后由证书管理器自动处理。

CMMF

请参阅 证书管理消息格式(CMMF)。

CRL

请参阅 证书撤销列表(CRL)。

CRMF

请参阅 证书请求消息格式(CRMF)。

cross-certification

证书交换，由两个不同的认证层次结构或链的两个 CA 交换。跨认证集扩展了信任链，涵盖这两个层次结构。另请参阅 证书颁发机构(CA)。

cryptographic algorithm

用于执行加密操作的规则或指示集合，如 encryption 和 解密。

CSP

请参阅 加密服务供应商(CSP)。

信任链

请参阅 证书链。

加密服务供应商(CSP)

代表使用标准接口（如 PKCS #11 定义）使用标准接口（如 PKCS #11 定义）的软件，用来执行加密服务（如密钥生成、密钥存储和加密）的加密模块。

加密模块

请参阅 PKCS #11 模块。

加密消息语法(CS)

用于数字签名、摘要、身份验证或加密任意消息的语法，如 CMMF。

基于证书的验证

基于证书和公钥加密的身份验证。另请参阅 基于密码的身份验证。

客户端 SSL 证书

用于使用 SSL 协议识别客户端到服务器的证书。请参阅 安全套接字层(SSL)。

客户端身份验证

将客户端识别到服务器的过程，如使用名称和密码，或使用证书以及一些数字签名的数据。请参阅 基于证书的验证、基于密码的身份验证、服务器身份验证。

证书扩展

X.509 v3 证书包含一个 extensions 字段，允许向证书添加任意数量的附加字段。证书扩展提供了向证书添加信息（如替代主题名称和使用量限制）的方法。很多标准扩展已经由 PKIX 工作组定义。

证书指纹

与证书关联的 单向哈希。数字不是证书本身的一部分，而是通过将 hash 功能应用到证书的内容来生成。如果证书的内容发生变化，即使单个字符也是如此，则同一功能也会生成不同的数字。因此，可以使用证书指纹来验证证书没有被篡改。

证书撤销列表(CRL)

根据 X.509 标准定义，由序列号（由 证书颁发机构(CA) 生成并签名）的撤销证书列表。

证书管理器

独立证书证书系统nbsp; 充当证书颁发机构的系统子系统。证书管理器实例问题、更新并撤销证书，该证书可与 CRL 一起发布至 LDAP 目录。它接受来自最终用户的请求。请参阅 证书颁发机构(CA)。

证书管理器代理

属于一组授权来管理证书管理器的代理服务的用户。这些服务包括访问和修改（批准和拒绝）证书请求和发布证书的功能。

证书管理消息格式(CMMF)

用于传达证书请求和撤销从最终用户到证书管理器的请求的消息格式，并将各种信息发送到最终实体。从互联网工程任务组(IETF)PKIX 工作组中提出标准。CMMF 被另一个建议的标准 通过 Cryptographic 消息语法(CMC)的证书管理消息 恢复。有关详细信息，请参阅 https://tools.ietf.org/html/draft-ietf-pkix-cmmf-02。

证书系统子系统

五个证书证书系统空间之一;系统经理： 证书管理器、在线证书状态管理器、密钥恢复授权 令牌密钥服务或令牌处理系统。

证书系统控制台

为任何单一证书证书系统nbsp;System 实例打开的控制台。CertificateCertificate Systemnbsp;System 控制台允许 CertificateCertificate Systemnbsp;System Administrator 控制对应证书证书系统证书系统nbsp;System 实例的配置设置。

证书请求消息格式(CRMF)

用于与 X.509 证书管理相关的消息的格式。这个格式是 CMMF 的子集。另请参阅 证书管理消息格式(CMMF)。有关详细信息，请参阅 https://tools.ietf.org/html/rfc2511。

证书配置集

组配置设置，用于定义特定类型的注册。证书配置集为特定类型的注册设置策略，以及证书配置集中的身份验证方法。

证书链

由连续证书颁发机构签名的证书的等级系列。CA 证书标识 证书颁发机构(CA)，用于签署该授权发布的证书。CA 证书可由父 CA 的 CA 证书注册，以此类推。Root CACertificateCertificate Systemnbsp;System 允许任何端点检索证书链中的所有证书。

证书颁发机构(CA)

验证证书要识别的个人或实体的身份后，一个可信实体会发出 certificate。CA 还更新并撤销证书并生成 CRL。证书在签发者字段中命名的实体始终是 CA。证书颁发机构可以独立使用证书认证服务器软件（如 Red Hat Certificate Systemnbsp）的独立第三方或机构 ;Hat Certificate Red Hat Certificate Systemnbsp;System.

跨对证书

由一个 CA 向另一个 CA 发布的证书，然后由两个 CA 存储以组成一个信任圆形。两个 CA 互相发布证书，然后将这两个证书存储为证书对。

通过 Cryptographic 消息语法(CMC)的证书管理消息

用于向证书管理器发出请求的消息格式。从互联网工程任务组(IETF)PKIX 工作组中提出标准。有关详细信息，请参阅 https://tools.ietf.org/html/draft-ietf-pkix-cmc-02。

链 CA

请参阅 链接的 CA。

D

delta CRL

包含自最后完整 CRL 开始已撤销的证书的 CRL 列表。

digital ID

请参阅 certificate。

双密钥对

两个公钥对，四个密钥完全对应于两个独立的证书。一个对的私钥用于签名操作，其他对的公钥和私钥用于加密和解密操作。每个对对应一个独立 certificate。另请参阅 加密密钥、public-key cryptography、签名密钥。

发行点

用于 CRL 来定义一组证书。每个发布点由一组发布的证书定义。可以为特定发布点创建 CRL。

可分辨名称(DN)

一系列 AVAs 来标识证书的主题。请参阅 属性值断言(AVA)。

密钥恢复授权

可选的独立证书证书系统板;System 子系统，用于管理用于最终实体的 RSA 加密密钥的长期归档和恢复。在发布新证书前，可以将证书管理器配置为通过密钥恢复授权归档最终实体的加密密钥。只有当最终实体（如敏感电子邮件）加密数据时，密钥恢复授权才很有用，该组织可能需要恢复一些天的时间。它只能与支持双键对的最终用户使用：两个单独的密钥对，一个用于加密，另一个用于数字签名。

密钥恢复授权中心存储密钥

在密钥恢复授权使用密钥恢复授权密钥解密后，由密钥恢复授权机构用来加密最终用户的加密密钥。存储密钥永不会离开密钥恢复授权。

密钥恢复授权代理

属于一组授权来管理密钥恢复机构的代理服务的用户，包括使用基于 HTML 的管理页面管理请求队列和授权恢复操作。

密钥恢复授权传输证书

认证由终端实体使用的公钥加密该实体的加密密钥，以传输到密钥恢复授权。密钥恢复授权使用与认证公钥对应的私钥来解密最终用户的密钥，然后使用存储密钥加密它。

密钥恢复机构恢复代理

那些拥有部分存储密钥的 n 人之一 密钥恢复授权。

数字签名

要创建数字签名，签名软件首先会从要签名的数据创建一个 单向哈希，比如新发布的证书。然后，使用签名者的私钥加密单向哈希。得到的数字签名对于每个数据已签名来说是唯一的。即使在消息中添加一个逗号也会更改该消息的数字签名。与签名者的公钥成功解密数字签名，并与相同数据的另一个哈希进行比较 tamper 检测。对包含公钥的证书验证 证书链 可提供签名者的身份验证。另请参阅 nonrepudiation、encryption。

解密

解放已加密的数据。请参阅 encryption。

E

eavesdropping

对通过接收不打算信息的实体通过网络发送的信息进行破坏。

Elliptic Curve Cryptography(ECC)

一个加密算法，使用 elliptic curves 为加密密钥基础的数学问题创建附加日志变量。ECC 密码比 RSA 密码更有效，由于其复杂性比 RSA 密码更强大，因此比 RSA 密码更强大。

encryption

以推断其含义的方式生成信息。请参阅 解密。

enrollment

请求和接收 X.509 证书以用于 公钥基础架构(PKI) 的过程。也称为 注册。

extensions 字段

请参阅 证书扩展。

加密密钥

仅用于加密的私钥。加密密钥及其对应的公钥以及 签名密钥 及其对等的公钥构成 双密钥对。

端点实体

在 公钥基础架构(PKI) 中，一个使用 certificate 的人、路由器、服务器或其他实体识别其自身。

F

fingerprint

请参阅 证书指纹。

FIPS PUBS 140

联邦信息标准公共(FIPS PUBS)140 是加密模块实施的美国政府标准，用于加密和解密数据的硬件或执行其他加密操作，如创建或验证数字签名。给美国政府的很多产品必须符合一个或多个 FIPS 标准。请参阅 http://www.nist.gov/itl/fipscurrent.cfm。

firewall

在两个或多个网络之间强制边界的系统或组合。

联邦网桥证书颁发机构(FBCA)

一个配置，其中两个 CA 组成一个信任方，方法是向彼此发出跨对证书，并将两个跨对证书存储为一个证书。

I

impersonation

作为通过网络发送的信息的预期接收者执行。模拟器可以采用两种形式： 欺骗 和 misrepresentation。

IP 欺骗

客户端 IP 地址的需要。

中间 CA

证书位于 证书链 中 root CA 和颁发的证书之间的 CA。

输入

在证书配置集功能的上下文中，它定义了特定证书配置集的注册表单。设定了每个输入，然后根据为此注册配置的所有输入动态创建注册表单。

J

JAR 文件

根据 Java™ 归档(JAR)格式 压缩的文件集合的数字信封。

Java™ Cryptography Architecture(JCA)

由 Sun Microsystems 为加密服务开发的 API 规格和引用。请参阅 http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.Introduction。

Java™ 原生接口(JNI)

标准接口在指定平台上提供不同实施 Java™ 虚拟机(JVM)的二进制兼容性，允许以 C 或 C++ 等语言编写的现有代码，以便统一平台绑定到 Java™。请参阅 http://java.sun.com/products/jdk/1.2/docs/guide/jni/index.html。

Java™ 安全服务(JSS)

用于控制网络安全服务(NSS)执行的安全操作的 Java™ 接口。

Java™ 开发套件(JDK)

由 Sun Microsystems 提供的软件开发套件，用于开发使用 Java™ 编程语言的应用程序和应用程序。

Java™ 归档(JAR)格式

将数字签名、安装程序脚本和其他信息与目录中的文件相关联的一组约定。

K

KEA

请参阅 主要交流算法(KEA)。

key

cryptographic algorithm 用于加密或解密数据的大量数字。例如，个人的 public key 允许其他人员为这个人加密信息。然后，信息必须使用对应的 私钥 来解密。

主要交流算法(KEA)

供美国政府用于密钥交换的算法。

密钥交换

遵循客户端和服务器来确定在 SSL 会话过程中将使用的对称密钥。

L

轻量级目录访问协议(LDAP)

一个目录服务协议，旨在在多个平台间通过 TCP/IP 运行。LDAP 是目录访问协议(DAP)的简化版本，用于访问 X.500 目录。LDAP 受 IETF 更改控制，并已演变为满足互联网要求。

链接的 CA

内部部署的 证书颁发机构(CA)，其证书由公共第三方 CA 签名。内部 CA 充当其问题的证书的 root CA，第三方 CA 充当链接到同一第三方 root CA 的其他 CA 发布的证书。也称为 "chained CA" 以及不同公共 CA 使用的其他术语。

M

MD5

由 Ronald Rivest 开发的消息摘要算法。另请参阅 单向哈希。

misrepresentation

将实体的表示为不是个人或组织。例如，当网站真正使用信用卡支付但从未发送任何货物的站点时，网站可能假定成为发放。Misrepresentation 是 impersonation 的一种形式。另请参阅 欺骗。

手动身份验证

配置证书证书系统nbsp 的方法；需要人工批准每个证书请求的系统子系统。使用这种身份验证方式时，servlet 在成功验证模块处理后将证书请求转发到请求队列。然后，在配置集处理和证书可以继续前，具有适当权限的代理必须单独批准每个请求。

消息摘要

请参阅 单向哈希。

N

non-TMS

非令牌管理系统。指的是配置 无法直接 处理智能卡的子系统（CA 以及可选的 KRA 和 OCSP）。

参见令牌管理系统(TMS).

nonrepudiation

无法由消息的发件人拒绝发送消息。数字签名 提供了一个非验证形式。

网络安全服务(NSS)

一组库，用于支持支持安全通信应用程序的跨平台开发。使用 NSS 库构建的应用程序支持 安全套接字层(SSL) 协议进行身份验证、tamper 检测和加密功能，以及用于加密令牌接口的 PKCS #11 协议。NSS 也作为软件开发套件单独提供。

O

OCSP

在线证书状态协议.

operation

在访问控制指令中允许或拒绝的特定操作，如读取或写入。

output

在证书配置集功能的上下文中，它从成功注册特定证书配置文件时，定义结果形式。每个输出都会被设置，然后动态地从为此注册配置的所有输出动态创建表单。

单向哈希

1. 很多固定长度从任意长度的数据生成，并附带哈希算法。数字也称为消息摘要，对散列数据是唯一的。数据更改（即使删除或更改单个字符）都会获得不同的值。

2.散列数据的内容无法从哈希中分离。

对象签名

一种文件签名方法，允许软件开发人员为 Java 代码、JavaScript 脚本或任何类型的文件进行签名，并允许用户识别信号，并通过签名的代码来控制对本地系统资源的访问。

对象签名证书

关联的私钥用于为对象签名的证书；与 对象签名 相关。

P

PKCS #10

公钥加密标准管理证书请求。

PKCS #11

管理加密令牌的公钥加密标准，如智能卡。

PKCS #11 模块

通过 PKCS #11 接口提供加密服务的加密设备的驱动程序，如加密和解密。PKCS #11 模块（也称为 加密模块或加密服务提供商 ）可以在硬件或软件中实施。PKCS #11 模块总是有一个或多个插槽，可以以某种形式的物理读取形式实施为物理硬件插槽，比如用于智能卡或软件中概念的插槽。PKCS #11 模块的每个插槽可以依次包含一个令牌，这是实际提供加密服务并选择性地存储证书和密

钥的硬件或软件设备。Red Hatnbsp;Hat 提供了一个内置的 PKCS #11 模块，它带有证书Certificate Systemnbsp;System.

PKCS #12

公钥加密标准管理密钥可移植性。

PKCS #7

管理签名和加密的公钥加密标准。

public key

在公钥加密中使用的一对密钥之一。公钥可以自由发布，并作为 certificate 的一部分发布。它通常用来加密发送到公钥所有者的数据，然后使用对应的 私钥 解密数据。

public-key cryptography

组成熟的技术和标准，允许实体以电子方式验证其身份或加密电子数据。两个密钥均涉及一个公钥和私钥。public key 作为证书的一部分发布，该证书将该密钥与特定身份关联。对应的私钥是保密的。使用公钥加密的数据只能使用该私钥解密。

公钥基础架构(PKI)

便于在网络环境中使用公钥加密和 X.509 v3 证书的标准和服务。

基于密码的身份验证

通过名称和密码确认身份。另请参阅 身份验证、基于证书的验证。

概念验证(POA)

使用私钥恢复认证机构传输密钥签名的数据，其中包含存档最终密钥的信息，包括密钥序列号、密钥恢复机构的名称、对应证书 主题名称 以及归档日期。签名的概念验证数据是密钥恢复授权机构在成功密钥存档操作后向证书管理器返回的响应。另请参阅 密钥恢复授权传输证书。

私钥

在公钥加密中使用的一对密钥之一。私钥被保留，用于解密使用对应 public key 加密的数据。

R

RC2, RC4

由 Rivest 为 RSA 数据安全而开发的密码算法。另请参阅 cryptographic algorithm。

RedRed Hat Certificate Systemnbsp;Hat CertificateRed Hat Certificate Systemnbsp;System

一组高度可配置的软件组件和工具，用于创建、部署和管理证书。CertificateCertificate Systemnbsp;System 由五个主要子系统组成，这些子系统可以安装在不同的证书证书系统空间中；位于不同物理位置的System 实例： 证书管理器、在线证书状态管理器、密钥恢复授权 令牌密钥服务和令牌处理系统。

Root CA

在证书链的顶部带有自签名证书的 证书颁发机构(CA)。另请参阅 CA 证书、subordinate CA。

RSA 密钥交换

SSL 的密钥更改算法，基于 RSA 算法。

RSA 算法

缩写为 Rivest-Shamir-Adleman，它是一个用于加密和身份验证的公共密钥算法。它是由 Ronald Rivest、Adi Shamir 和 Leonard Adleman 开发，于 1978 年推出。

注册

请参阅 enrollment。

S

sandbox

针对 Java™ 代码必须操作的明确定义限制的 Java™ 术语。

servlet

代表证书证书系统nbsp;System 子系统，处理特定类型与最终用户交互的 Java™ 代码。例如，证书注册、撤销和密钥恢复请求各自由单独的 servlet 处理。

SHA

安全散列算法，由美国政府使用的哈希功能。

slot

PKCS #11 模块 的一部分，在包含 token 的硬件或软件中实施。

SSL

请参阅 安全套接字层(SSL)。

subject

由 certificate 标识的实体。特别是，证书的 subject 字段包含一个 主题名称，它唯一描述了已认证实体。

subordinate CA

该证书由另一个从属 CA 或由 root CA 签名的证书颁发机构。请参阅 CA 证书、Root CA。

主题名称

可分辨名称(DN) 唯一描述了 subject 的 certificate。

单点登录

1. 在 CertificateCertificate Systemnbsp;System 中，该密码简化了到 Red Hat Certificate Systemnbsp 签名的密码;Hat Certificate Red Hat Certificate Systemnbsp;Hat Certificate Red Hat Certificate Systemnbsp;System by store password for internal database and tokens.每次用户登录时，都需要输入此单一密码。

2.用户一次登录单个计算机，并由网络中的各种服务器自动进行身份验证。部分单点登录解决方案采用多种形式，包括自动跟踪用于不同服务器的密码的机制。证书支持 公钥基础架构(PKI) 中的单点登录。用户可以一次登录本地客户端的私钥数据库，只要客户端软件正在运行，则依靠 基于证书的验证 来访问该用户的机构中的每个服务器。

安全域

PKI 子系统的集中存储库或清单。其主要目的是通过在子系统之间自动建立可信关系来促进新 PKI 服务的安装和配置。

安全套接字层(SSL)

允许在客户端和服务器间进行 mutual身份验证的协议，并建立经过身份验证的和加密连接。SSL 在 TCP/IP 和以下 HTTP、LDAP、IMAP、NNTP 和其他高级别网络协议上运行。

安全频道

TPS 和智能卡间的安全关联，允许基于 TKS 和智能卡 APDU 生成的共享主密钥加密。

对称加密

使用同一加密密钥来加密和解密给定消息的加密方法。

智能卡

包含微处理器并存储加密信息（如密钥和证书）的小型设备，并执行加密操作。智能卡实现一些或所有 PKCS #11 接口。

服务器 SSL 证书

用来使用 安全套接字层(SSL) 协议将服务器标识到客户端的证书。

服务器身份验证

识别服务器到客户端的进程。另请参阅 客户端身份验证。

欺骗

刚开始成为其他人.例如，个人可以预告具有电子邮件地址 jdoe@example.com，或者计算机可以在不能识别为名为 www.redhat.com 的网站。欺骗是 impersonation 的一种形式。另请参阅 misrepresentation。

签名密钥

仅用于签名的私钥。签名密钥及其等效公钥以及 加密密钥 及其对等公钥，构成 双密钥对。

签名的审计日志

请参阅 审计日志。

签名算法

用于创建数字签名的加密算法。CertificateCertificate Systemnbsp;System 支持 MD5 和 SHA 签名算法。另请参阅 cryptographic algorithm、数字签名。

签名证书

公钥与用于创建数字签名的私钥对应的证书。例如，证书管理器必须具有其公钥与用于为其发出的证书签名的私钥的签名证书。

自我测试

测试证书证书系统空间的功能；当实例启动时和按需启动时，System 实例都会启动和按需。

T

tamper 检测

确保以电子形式接收的数据的机制与同一数据的原始版本完全对应。

token

与 slot 中的 PKCS #11 模块 关联的硬件或者软件设备。它提供了加密服务，并选择性地存储证书和密钥。

trust

信心依赖个人或其他实体。在 公钥基础架构(PKI) 中，信任指的是证书的用户和签发证书的 证书颁发机构(CA) 之间的关系。如果 CA 受信任，则 CA 发布的有效证书可以被信任。

令牌处理系统(TPS)

直接与企业安全客户端和智能卡交互的子系统，以管理这些智能卡上的密钥和证书。

令牌密钥服务(TKS)

令牌管理系统中的子系统，根据智能卡 APDU 和其他共享信息为每个智能卡生成特定的密钥，如令牌 CUID。

令牌管理系统(TMS)

相互相关的子系统 - CA、TKS、TPS 以及可选的 KRA，用于管理智能卡(token)上的证书。

索引

证书管理器

administrators

创建，创建用户

代理

创建，创建用户

发布目录的手动更新，更新目录中的证书和 CRL

密钥对和证书

CA 签名证书，CA 签名密钥对和证书

OCSP 签名证书，OCSP 签名密钥对和证书

SSL 服务器证书，SSL 服务器密钥对和证书

TLS CA 签名证书，OCSP 签名密钥对和证书

子系统证书，子系统证书

序列号范围，在签发者证书上更改 CA 的限制

配置

通知的 SMTP 设置，为证书证书系统配置邮件服务器; 系统通知

证书系统

备份，备份和恢复证书证书证书nbsp;系统

恢复，备份和恢复实例目录

证书系统控制台

Status 选项卡，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS Subsystems

管理日志，在控制台中查看日志

配置标签页，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS Subsystems

配置身份验证，设置基于目录的身份验证，设置基于 PIN 的注册

证书系统数据

它存储在的位置，配置 LDAP 数据库

证书续订，配置配置集以启用续订

证书设置向导

使用 安装证书，通过控制台安装证书

使用 安装证书链，通过控制台安装证书

证书配置集

签名算法，为证书设置签名算法

U

users

创建，创建用户

附录 F. 修订历史记录

请注意，修订号与该手册版本相关，而不是与红帽认证系统的版本号相关。

修订 9.7-0 **Tue Sep 29 2020** **Florian Delehaye**

指南的 Red Hat Certificate System 9.7 发行版本。

修订 9.6-0 **Wed Sep 16 2020** **Florian Delehaye**

指南的 Red Hat Certificate System 9.6 发行版本。大量修订：更新信息并在整个指南中重新组织内容。

修订 9.5-0 **Tue Aug 06 2019** **Marc Muehlfeld**

指南的 Red Hat Certificate System 9.5 发行版本。

修订 9.4-0 **Thu Oct 25 2018** **Marc Muehlfeld**

指南的 Red Hat Certificate System 9.4 发行版本。

修订 9.3-1 **Thu May 03 2018** **Marc Muehlfeld**

添加了使用 CMC 发布的证书、执行 CMC Revocation 以及用于 CMC 配置 的证书 的一些增强功能。

修订 9.3-0 **Tue Apr 10 2018** **Marc Muehlfeld**

对于版本 9.3：添加多个部分，如 KRA 操作加密、更新系统证书，并使用不同的 SCP 版本。

修订 9.2-0 ***Tue Aug 01 2017*** ***Petr Bokoč***

Red Hat Certificate System 9.2 GA release

修订 9.1-1 ***Thu Mar 09 2017*** ***Petr Bokoč***

异步更新

修订 9.1-0 ***Mon Aug 15 2016*** ***Petr Bokoč***

Red Hat Certificate System 9.1 release

修订 9.0-0 ***Mon Aug 15 2016*** ***Petr Bokoč***

Red Hat Certificate System 9.0 的初始版本。