Red Hat Certificate System 10

管理指南

为 Red Hat Certificate System 10.4 更新

Last Updated: 2023-09-23

Red Hat Certificate System 10 管理指南

为 Red Hat Certificate System 10.4 更新

Florian Delehaye
Red Hat Customer Content Services
fdelehay@redhat.com

Marc Muehlfeld
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Filip Hanzelka
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律通告

摘要

本手册涵盖了安装、配置和管理证书系统子系统的所有方面。它还涵盖了管理任务，如添加用户；请求、续订和撤销证书；发布 CRL；以及管理智能卡。本指南适用于证书系统管理员。

目录

第 1 章 RED HAT CERTIFICATE SYSTEM 子系统概述 6
1.1. 证书使用 6
1.2. 证书系统子系统的审阅 6
1.3. A LOOK AT MANAGE CERTIFICATES (NON-TMS) 6
1.4. 令牌管理系统(TMS)的 LOOK 6
1.5. RED HAT CERTIFICATE SYSTEM 服务 7

部分 I. RED HAT CERTIFICATE SYSTEM 用户界面 8

第 2 章 用户界面 9
2.1. 用户界面概述 9
2.2. 客户端 NSS 数据库初始化 9
2.3. 图形界面 10
2.4. WEB 界面 11
2.5. 命令行界面 17
2.6. 企业安全客户端 23

部分 II. 设置证书服务 26

第 3 章 为颁发证书（证书配置文件）进行规则 27
3.1. 关于证书配置文件 27
3.2. 设置证书配置文件 31
3.3. 在配置集中定义密钥默认值 45
3.4. 配置配置集以启用续订 46
3.5. 为证书设置签名算法 47
3.6. 管理 CARELATED 配置集 50
3.7. 管理主题名称和主题备用名称 60

第 4 章 设置 KEY ARCHIVAL 和恢复 68
4.1. 在控制台中配置代理应用密钥恢复 69
4.2. 测试密钥 ARCHIVAL 和恢复设置 69

第 5 章 请求、注册和管理证书 72
5.1. 关于注册和续订证书 72
5.2. 创建证书签名请求 72
5.3. 请求和接收证书 85
5.4. 续订证书 90
5.5. 使用 CMC 提交证书请求 95
5.6. 执行 BULKSUANCE 114
5.7. 在 CISCO 路由器上注册证书 117
5.8. 使用证书转换 125

第 6 章 使用和配置令牌管理系统：TPS 和 TKS 128
6.1. TPS 配置集 128
6.2. TPS 操作 128
6.3. 令牌策略 130
6.4. 令牌操作和策略处理 132
6.5. 内部注册 140
6.6. 外部注册 141
6.7. 映射解决程序配置 146
6.8. 身份验证配置 150
6.9. 连接器 151
6.10. 吊销路由配置 153

6.11. 设置服务器端密钥生成 153
6.12. 设置新密钥集 156
6.13. 设置新主密钥 157
6.14. 设置 TKS/TPS 共享 SYMMETRIC 密钥 161
6.15. 对不同的 SCP 版本使用不同的 APPLETS 164

第 7 章 吊销证书并颁发 CRL **167**
7.1. 关于撤销证书 167
7.2. 执行 CMC 吊销 171
7.3. 发布 CRL 176
7.4. 设置 FULL 和 DELTA CRL 计划 187
7.5. 启用撤销检查 192
7.6. 使用在线证书状态协议(OCSP)恢复器 192

第 8 章 管理 PKI ACME RESPONDER **211**
8.1. 启用/禁用 ACME 服务 211
8.2. 检查 PKI ACME RESPONDER 的状态 211

部分 III. 管理 CA 服务的其他配置 **213**

第 9 章 发布证书和 CRL **214**
9.1. 关于发布 214
9.2. 配置发布到文件 218
9.3. 配置发布到 OCSP 221
9.4. 配置发布到 LDAP 目录 224
9.5. 创建规则 231
9.6. 启用发布 236
9.7. 启用发布队列 238
9.8. 设置可恢复的 CRL 下载 240
9.9. 发布跨证书 240
9.10. 测试发布到文件 242
9.11. 查看证书和 CRL 发布到文件 244
9.12. 更新目录中的证书和 CRL 244
9.13. 注册自定义映射程序和过期插件模块 246

第 10 章 注册证书的身份验证 **249**
10.1. 配置代理应用注册 249
10.2. 自动注册 250
10.3. CMC 身份验证插件 263
10.4. CMC SHAREDSECRET 身份验证 265
10.5. 测试注册 267
10.6. 注册自定义身份验证插件 268
10.7. 使用命令行手动检查证书状态 271
10.8. 使用 WEB 界面手动检查证书状态 271

第 11 章 注册证书的授权(ACCESS EVALUATORS) **273**
11.1. 授权机制 273
11.2. 默认评估 273

第 12 章 使用自动通知 **275**
12.1. 关于 CA 的自动化通知 275
12.2. 为 CA 设置自动通知 276
12.3. 自定义通知消息 279
12.4. 为证书系统通知配置邮件服务器 283
12.5. 为 CA 创建自定义通知 284

第 13 章 设置自动化作业 **285**
13.1. 关于自动化作业 285
13.2. 设置作业调度程序 287
13.3. 设置特定作业 288
13.4. 注册作业模块 296

部分 IV. 管理子系统实例 **299**

第 14 章 基本子系统管理 **300**
14.1. PKI 实例 300
14.2. PKI 实例执行管理 301
14.3. 打开子系统控制台和服务 305
14.4. 在 JAVA 安全管理器下运行子系统 311
14.5. 配置 LDAP 数据库 312
14.6. 查看安全域配置 322
14.7. 管理用于子系统的 SELINUX 策略 323
14.8. 备份和恢复证书系统 326
14.9. 运行自测试 332

第 15 章 管理系统用户和组 **336**
15.1. 关于授权 336
15.2. 默认组 336
15.3. 管理 CA、IADP、KRA 或 TKS 的用户和组 340
15.4. 为 TPS 创建和管理用户 354
15.5. 为用户配置访问控制 361

第 16 章 配置子系统日志 **369**
16.1. 关于证书系统日志 369
16.2. 管理日志 374
16.3. 使用日志 386

第 17 章 管理子系统证书 **394**
17.1. 所需的子系统证书 394
17.2. 通过控制台请求证书 403
17.3. 续订子系统证书 425
17.4. 更改子系统证书的名称 429
17.5. 使用跨证书 433
17.6. 管理证书数据库 434
17.7. 更改 CA 证书的信任设置 443
17.8. 管理子系统使用的令牌 445

第 18 章 在 RED HAT ENTERPRISE LINUX 7 中设置时间和日期 **447**
更改当前时间 447
更改当前日期 447

第 19 章 确定证书系统产品版本 **448**

第 20 章 更新 RED HAT CERTIFICATE SYSTEM **449**

第 21 章 故障排除 **450**

第 22 章 子系统控制和维护 **455**
22.1. 启动、停止、重启和获取状态 455
22.2. 子系统健康检查 456

部分 V. 参考 **460**

附录 A. 证书配置文件输入和输出参考 461
A.1. 输入参考 461
A.2. 输出参考 467

附录 B. 证书和 CRL 的默认值、约束和扩展 469
B.1. 默认参考 469
B.2. 约束参考 520
B.3. 标准 X.509 V3 证书扩展参考 531
B.4. CRL 扩展 544

附录 C. 发布模块参考 564
C.1. 发布程序插件模块 564
C.2. 映射程序插件模块 568
C.3. 规则实例 575

附录 D. ACL 参考 578
D.1. 关于 ACL 配置文件 578
D.2. 常见 ACL 579
D.3. 特定于证书的 ACL 585
D.4. 密钥恢复特定于授权的 ACL 597
D.5. 特定于在线证书状态管理器的 ACL 602
D.6. 令牌密钥服务特定 ACL 606

附录 E. 审计事件 609
E.1. 审计事件描述 609

术语表 622

索引 640

附录 F. 修订历史记录 658

第 1 章 RED HAT CERTIFICATE SYSTEM 子系统概述

每个常见的 PKI 操作 - 发布、续订和撤销证书；归档和恢复密钥；发布 CRL 并验证证书状态 - 通过 Red Hat Certificate System 中的操作子系统实现。本章介绍了每个子系统的功能及其一起建立强大和本地 PKI 的方法。

1.1. 证书使用

证书的目的是建立信任。它们的使用情况因用于确保的信任类型而异。某些类型的证书用于验证显示者的身份；其他证书用于验证对象或项目未被篡改。

有关如何使用证书、证书类型或证书如何建立身份和关系的详情，请参考 *Red Hat Certificate System Planning, Installation, and Deployment Guide* 中的 *Certificates and Authentication* 部分。

1.2. 证书系统子系统的审阅

Red Hat Certificate System 提供五个不同的子系统，每个子系统侧重于 PKI 部署的不同方面。这些子系统协同工作来创建公钥基础架构(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 *Red Hat Certificate System Planning、安装和部署指南中的证书系统子系统 部分*。

企业安全客户端
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。企业安全客户端是一个用户界面，允许人员轻松地在智能卡上管理证书。企业安全客户端将所有令牌操作（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，请参阅使用 *企业安全客户端管理智能卡。*

1.3. A LOOK AT MANAGE CERTIFICATES (NON-TMS)

传统 PKI 环境提供基本的框架来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TMS 环境也可以使用 OCSP 响应者 和 KRA 实例。

有关此主题的详情，请查看 *Red Hat Certificate System Planning, Installation, installation, and Deployment Guide* 中的以下部分：

- 管理证书
- 使用单个证书管理器
- Lost 密钥规划：key Archival 和 Recovery
- 平衡证书请求处理
- 平衡客户端 OCSP 请求

1.4. 令牌管理系统(TMS)的 LOOK

证书系统创建、管理、续订和吊销证书，同时还存档和恢复密钥。对于使用智能卡的机构，证书系统具有令牌管理系统 - 带有已建立关系的子系统集合 - 生成密钥和请求并接收要用于智能卡的证书。

有关此主题的详情，请查看 *Red Hat Certificate System Planning, Installation, installation, and Deployment Guide* 中的以下部分：

- 使用智能卡(TMS)
- 使用智能卡

1.5. RED HAT CERTIFICATE SYSTEM 服务

根据用户类型，可以通过各种不同的接口来管理证书和子系统：管理员、代理、审核员和最终用户。有关通过每个接口执行的不同功能的概述，请参阅 *用户界面部分*。

部分 I. RED HAT CERTIFICATE SYSTEM 用户界面

第 2 章 用户界面

根据用户的角色，可以通过不同的接口来管理证书和子系统：管理员、代理、审核员和最终用户。

2.1. 用户界面概述

管理员可以使用以下界面与已完成的证书系统安装安全地交互：

- PKI 命令行界面和其他命令行工具
- PKI 控制台图形界面
- 证书系统 Web 界面。

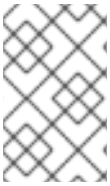
这些接口需要在使用之前配置，以通过 TLS 与证书系统服务器安全通信。不允许使用这些客户端而无需正确配置。其中一些工具使用 TLS 客户端身份验证。在需要时，其所需的初始化过程包括配置此功能。使用哪个界面取决于管理员的首选项和功能。本章后面介绍了使用这些接口的常见操作。

默认情况下，PKI 命令行界面使用用户的 **~/.dogtag/nssdb/** 目录中的 NSS 数据库。第 2.5.1.1 节 “pki CLI 初始化” 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 “使用 "pki" CLI” 中描述了使用 PKI 命令行工具的一些示例。在指南的其余部分中显示了其他示例。

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，与证书系统（作为其他用户角色中的管理员）交互。它们在 第 2.5 节 “命令行界面” 中进行了描述，如 第 2.5.2 节 “AtoB”。这些工具在以后的小节中会使用，如 第 5.2.1.2 节 “使用 **PKCS10Client**创建 CSR”。

证书系统 Web 界面允许通过 Firefox Web 浏览器进行管理访问。第 2.4.1 节 “浏览器初始化” 描述配置客户端身份验证的说明。第 2.4 节 “Web 界面” 中描述的其他部分使用证书系统的 Web 界面。

证书系统的 PKI 控制台是一个图形界面。**请注意，它已被弃用。** 第 2.3.1 节 “**pkiconsole** 初始化” 描述了如何初始化这个控制台界面。第 2.3.2 节 “将 **pkiconsole** 用于 CA、OCSP、KRA 和 TKS 子系统” 提供有关使用它的概述。后续部分，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文件” 为特定操作进行了更详细的信息。



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。命令行工具在执行操作并返回提示后立即终止自己，因此管理员部分不需要任何操作来终止会话。

2.2. 客户端 NSS 数据库初始化

在 Red Hat Certificate System 上，某些接口可能需要使用 TLS 客户端证书身份验证（双向身份验证）访问服务器。在执行服务器端管理任务前，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。它们可以在 NSS 数据库中生成，或者从其它位置导入，例如从 PKCS the 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 初始化”

- 第 2.3.1 节 “**pkiconsole** 初始化”
- 第 2.4.1 节 “浏览器初始化”

2.3. 图形界面

重要

pkiconsole 已被弃用。

证书系统控制台**pkiconsole** 是一个图形界面，专为具有管理员角色权限的用户而设计，以管理子系统本身。这包括添加用户、配置日志、管理配置集和插件以及许多其他功能。此实用程序使用客户端身份验证通过 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 初始化

要第一次使用 **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
```

此命令在 **~/.redhat-idm-console/** 目录中创建新客户端 NSS 数据库。

要将 CA 证书导入到 PKI 客户端 NSS 数据库中，请参阅 *Red Hat Certificate System Planning, Installation, and Deployment Guide* 中的*将证书导入到 NSS 数据库* 部分。

要请求新客户端证书，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
```

按照 *Red Hat Certificate System Planning, Installation, and Deployment Guide* 中的 *Managing Certificate/Key Crypto Token* 部分所述，验证并导入 admin 客户端证书：

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

重要

在导入 CA admin 客户端证书前，请确保所有中间证书和 root CA 证书都已导入。

要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 子系统

Java 控制台供四个子系统使用：CA、IADP、KRA 和 TKS。可以使用本地安装的 **pkiconsole** 工具访问控制台。它可以访问任何子系统，因为命令需要主机名、子系统的管理 TLS 端口和特定的子系统类型。

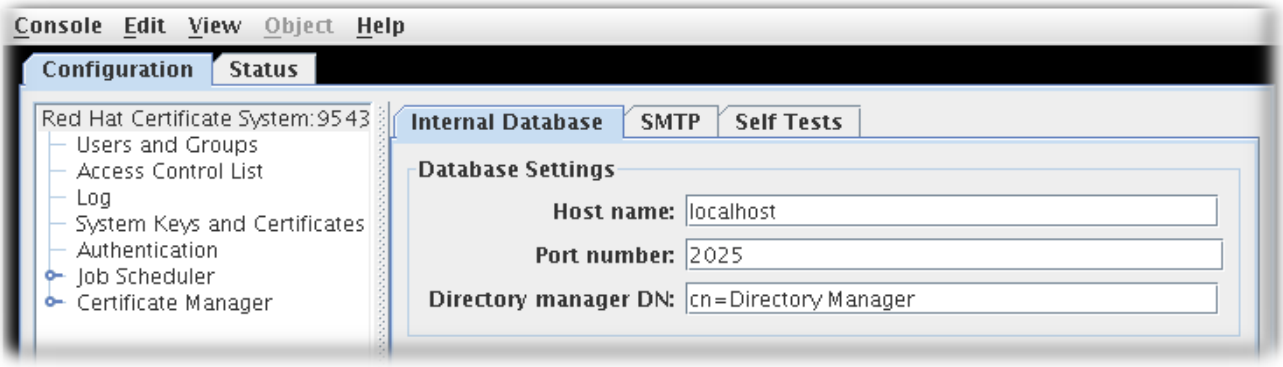
```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

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
```

这会打开控制台，如 图 2.1“证书系统控制台” 中所示。

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。此选项卡中可用的选项根据实例的子系统类型而有所不同；CA 具有最大选项，因为它具有对作业、通知和证书注册身份验证的额外配置。

所有子系统有四个基本选项：

- 用户和组
- 访问控制列表
- 日志配置
- 子系统证书（例如，在安全域或审计签名中）签发到子系统的证书。

Status 选项卡显示子系统维护的日志。

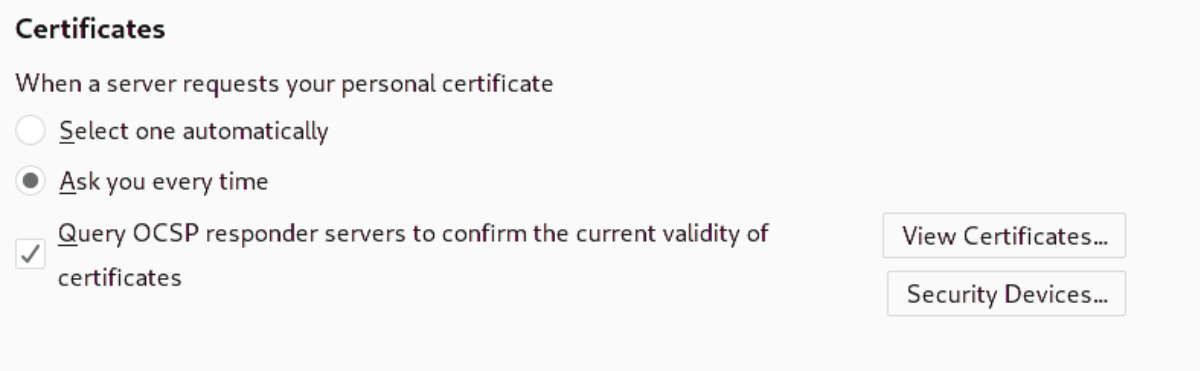
2.4. WEB 界面

2.4.1. 浏览器初始化

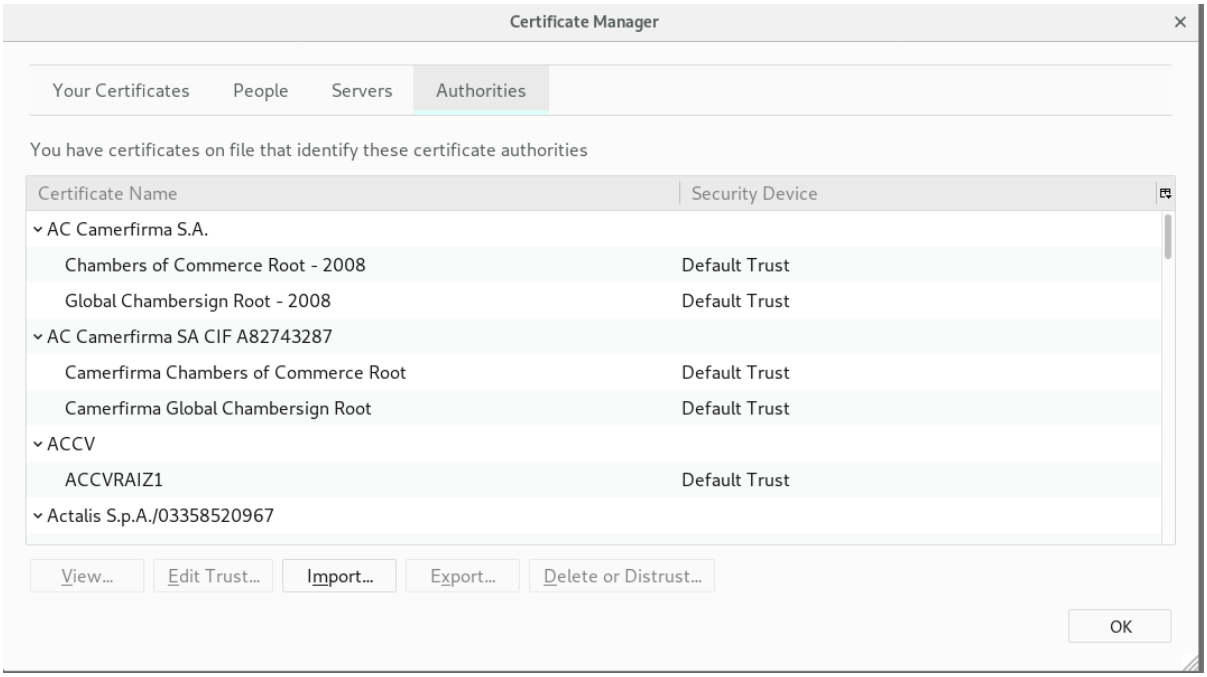
本节介绍 Firefox 访问 PKI 服务的浏览器初始化。

导入 CA 证书

1. 点 **Menu → Preferences → Privacy & Security → View certificate**。



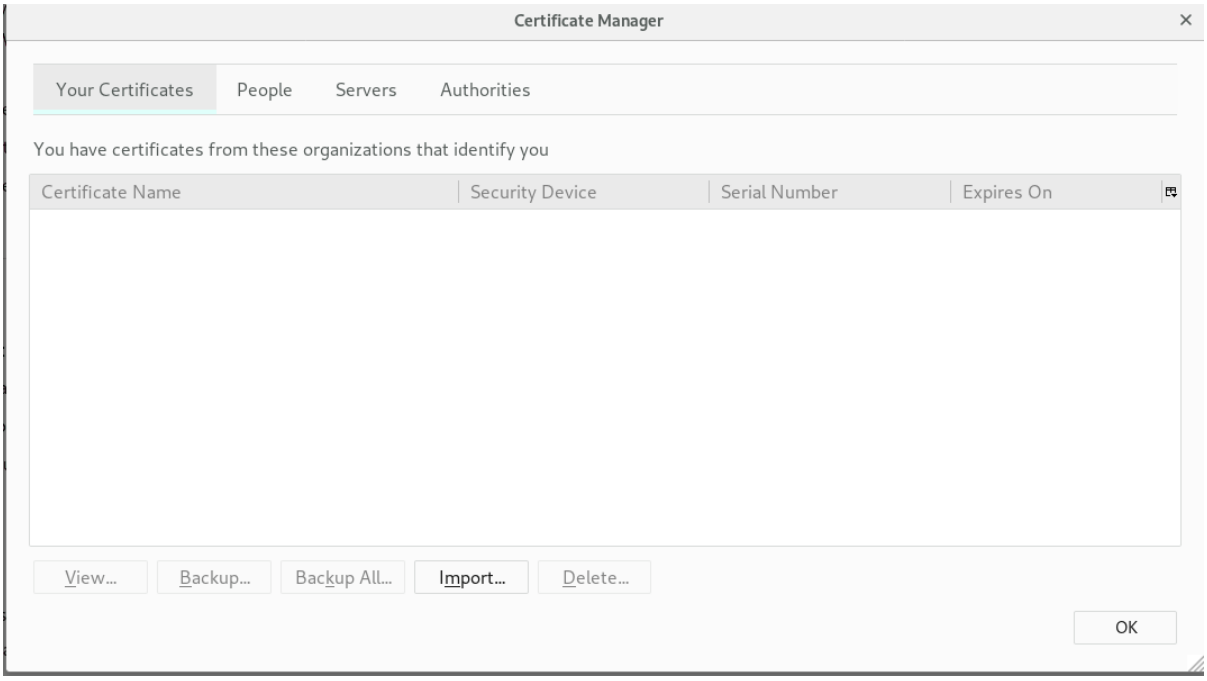
2. 选择 **"权限 "**选项卡，然后单击 **导入** 按钮。



3. 选择 **ca.crt** 文件并点击 **Import**。

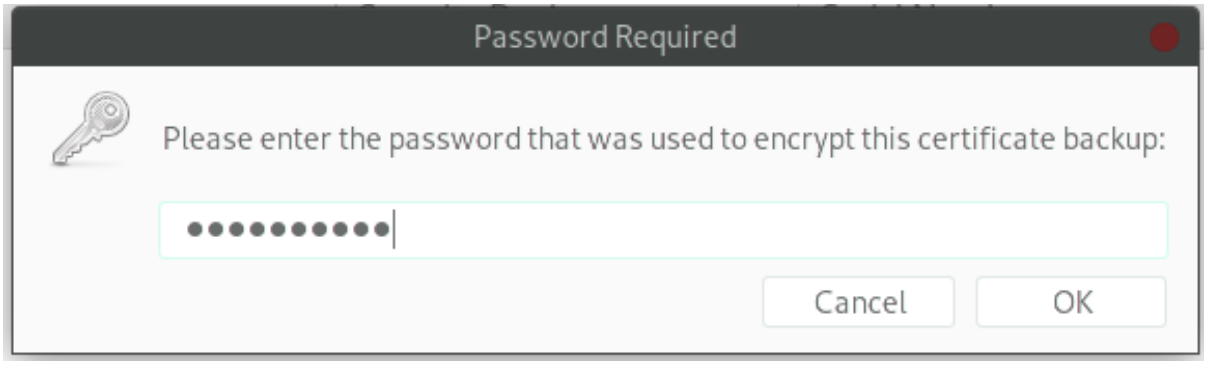
导入客户端证书

1. 点 **Options → Preferences → Privacy & Security → View certificate**。
2. 选择 **Your Certificates** 选项卡。



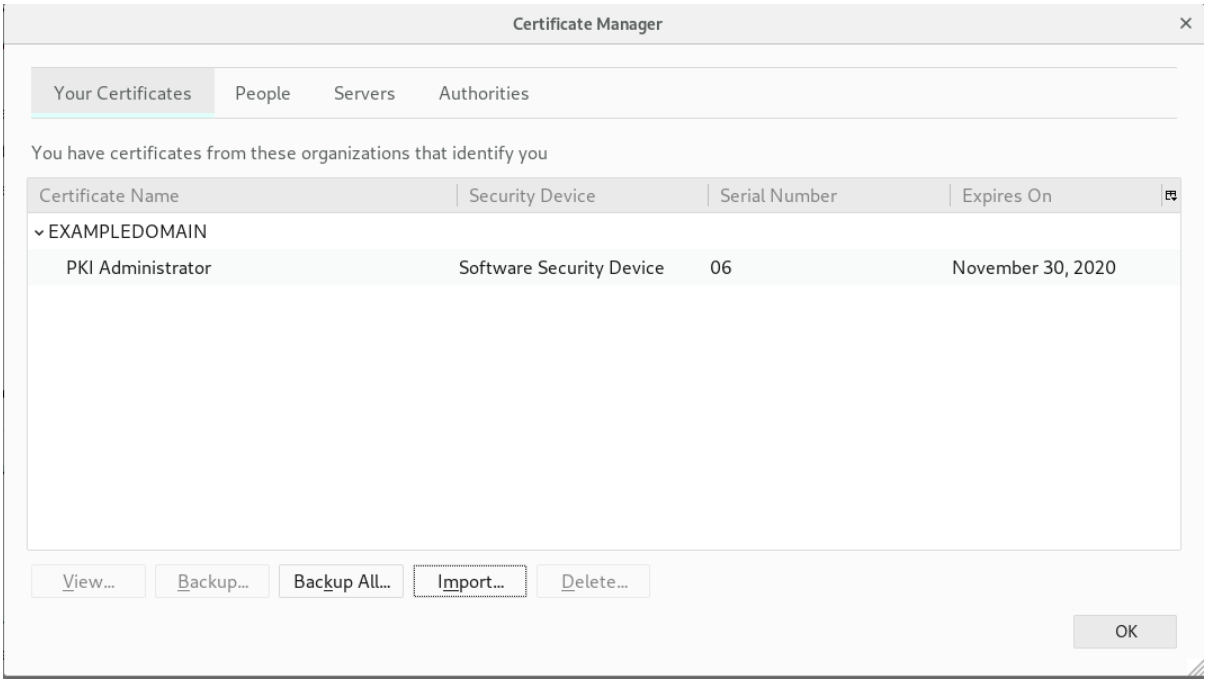
3. 单击 **Import**，再选择客户端 p12 文件，如 **ca_admin_cert.p12**。

4. 在提示符处输入客户端证书的密码。



5. 点**确定**。

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*:*port*** 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。可通过输入主机名和安全端口作为 URL 访问，使用管理员的证书进行身份验证，然后单击适当的 **Administrators** 链接。

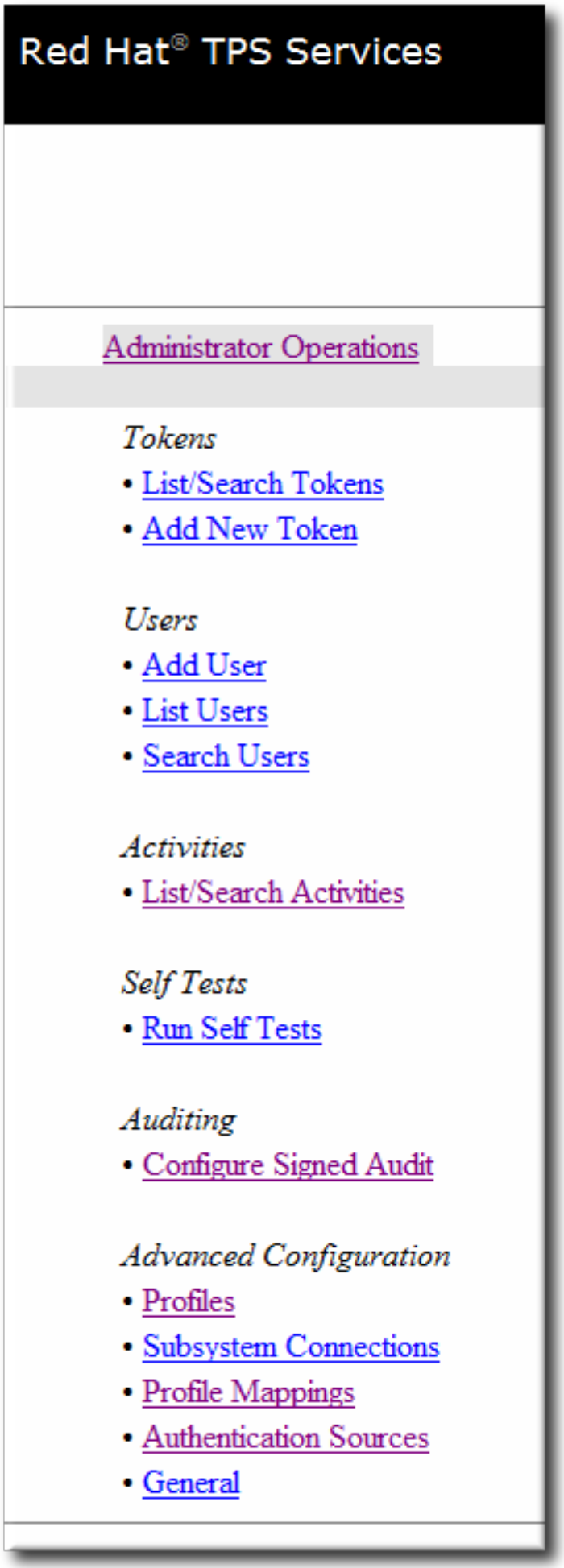
注意

所有子系统都有一个 TLS 端口，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 控制台更有限；主要管理功能是管理子系统用户。

TPS 仅允许操作管理 TPS 子系统的用户。但是，TPS 管理页面也可以列出令牌，并显示在 TPS 上执行的所有活动（包括通常隐藏的管理操作）。

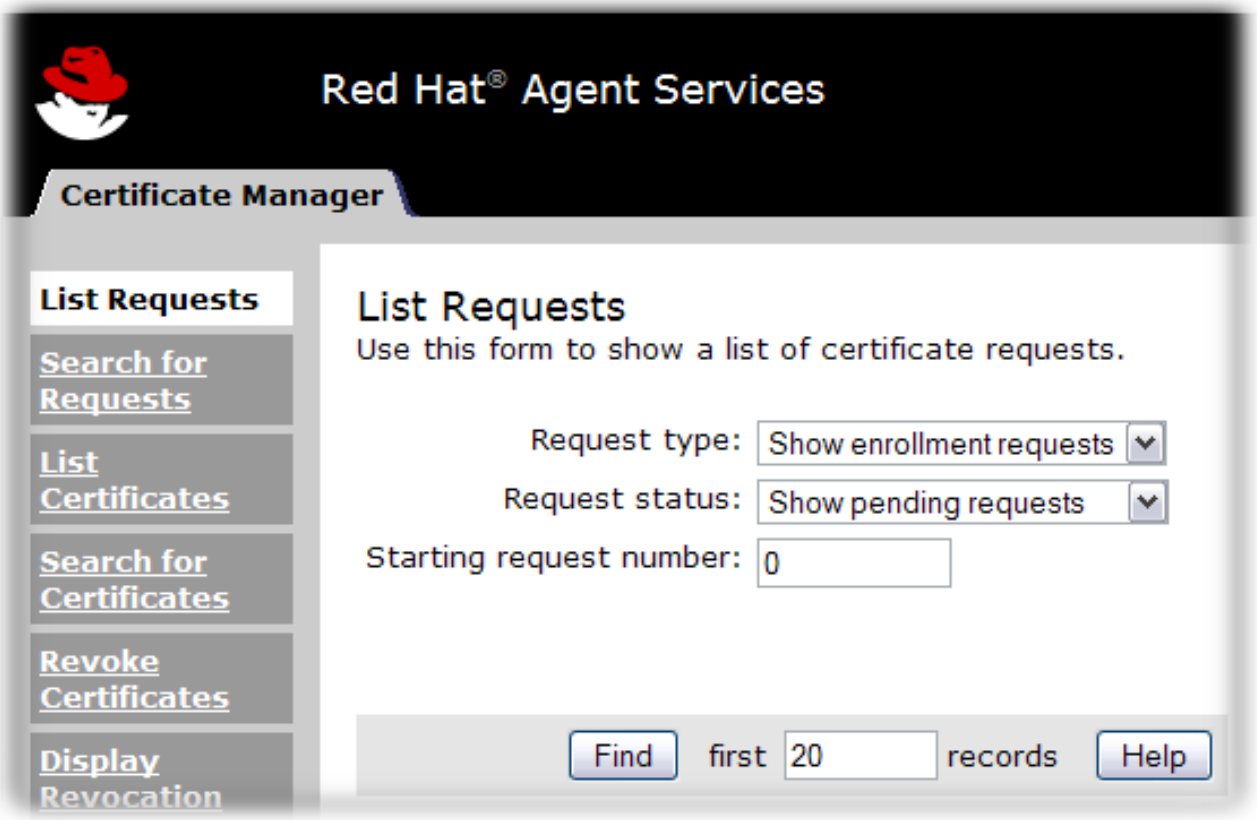
图 2.2. TPS Admin Page


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

代理服务页面几乎执行所有证书和密钥管理任务。这些服务基于 HTML，代理使用特殊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



具体操作因子系统而异：

- 证书管理器代理服务包括批准证书请求（签发证书）、撤销证书以及发布证书和 CRL。CA 发布的所有证书都可以通过其代理服务页面进行管理。
- TPS 代理服务（如 CA 代理服务）管理已格式化且已通过 TPS 向它们发布证书的所有令牌。令牌可以被代理注册、暂停和删除。另外两个角色(operator 和 admin)可在 Web 服务页面中查看令牌，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，如果证书丢失，该请求是否允许使用现有密钥对来签发证书。
- OCSP 代理服务页面允许代理配置将 CRL 发布到 OCSP 的 CA，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。

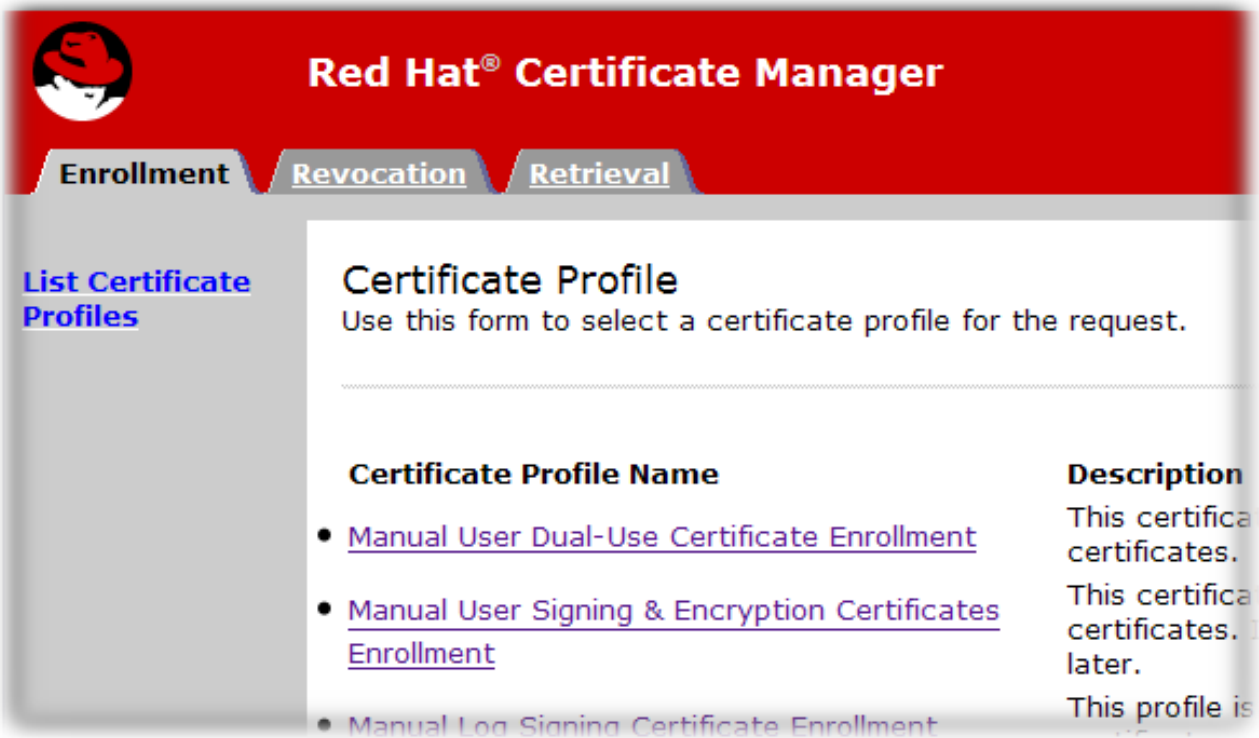
2.4.4. 最终用户页面

CA 和 TPS 都以某种方式处理直接用户请求。这意味着最终用户必须有一个方法来连接这些子系统。CA 具有 *最终用户或终端实体* 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；也可以使用服务器的主机名和特定的端到端 TLS 端口通过 HTTPS 访问它们。

对于 CA，每种类型的 TLS 证书都通过一个特定的在线提交表单进行处理，称为 *profile*。CA 有大约两个取消注册的证书配置文件，涵盖所有类型的证书 - 用户 TLS 证书、服务器 TLS 证书、日志和文件签名证书、电子邮件证书和每种类型的子系统证书。也可以有自定义配置集。

图 2.4. 证书管理器的端到端页面



最终用户在签发证书时通过 CA 页面检索其证书。它们还可以下载 CA 链和 CRL，并通过这些页面撤销或更新其证书。

2.5. 命令行界面

本节讨论命令行工具。

2.5.1. "pki" CLI

pki 命令行界面(CLI)使用 REST 界面提供对服务器上的各种服务的访问（请参阅 *Red Hat Certificate System Planning, Installation, installation, installation, and Deployment Guide* 中的 *REST Interface* 部分）。CLI 可以按如下方式调用：

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

请注意，必须在命令前放置 CLI 选项，并在命令后面使用命令参数。

2.5.1.1. pki CLI 初始化

要第一次使用命令行界面，请指定新密码并使用以下命令：

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

这将在 **~/.dogtag/nssdb** 目录中创建新客户端 NSS 数据库。在使用客户端 NSS 数据库的所有 CLI 操作中必须指定密码。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 数据库中，请参阅 *Red Hat Certificate System Planning, Installation, and Deployment Guide* 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 the 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
```

按照 *Red Hat Certificate System Planning, Installation, and Deployment Guide* 中的 *Managing Certificate/Key Crypto Token* 部分所述，验证并导入 admin 客户端证书：

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

重要

在导入 CA admin 客户端证书前，请确保所有中间证书和 root CA 证书都已导入。

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 the 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. 使用 "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，请执行 **pki** 命令，无需任何其他命令或参数：

```
$ pki
```

有些命令有子命令。要列出它们，请使用命令名称执行 **pki**，且不执行其他选项。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
```

要查看手册页，请指定命令行 **help** 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，请指定证书别名、客户端 NSS 数据库密码以及服务器 URL （可选）：

```
$ 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 在 **http://*local_host_name*:8080 与服务器通信**。要在不同位置与服务器通信，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 实用程序将 Base64 编码的证书解码成其二进制等效证书。例如：

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

详情请查看 AtoB(1) man page。

2.5.3. AuditVerify

auditVerify 工具通过在日志条目上验证签名来验证审计日志的完整性。

Example:

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

示例使用 **~jsmith/auditVerifyDir** NSS 数据库(**-d**)中的 **Log Signing Certificate** (**-n**)验证审计日志。要验证的日志列表(**-a**)位于 **~jsmith/auditVerifyDir/logListFile** 文件中，用逗号分开和排序。前面带有证书和密钥数据库文件名的前缀(**-P**)为空。输出为详细(**-v**)。

详情请查看 AuditVerify(1) man page 或 第 16.3.2 节 “使用签名的审计日志”。

2.5.4. BtoA

BtoA 实用程序使用 Base64 编码二进制数据。例如：

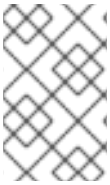
```
$ BtoA input.bin output.ascii
```

详情请查看 BtoA(1) man page。

2.5.5. CMCRequest

CMCRequest 工具创建一个证书颁发或撤销请求。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。有关配置文件选项以及详情，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**吊销证书” 请求和接收证书。

2.5.6. CMCRevoke

Legacy。不要使用。

2.5.7. CMCSharedToken

CMCSharedToken 工具为共享 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**)使用名为 **subsystemCert cert-pki-tomcat** (**-n**) 的 **NSS** 数据库中的证书进行加密，并存储在 **cmcSharedtok2.b64** 文件(**-o**)中。默认安全令牌 **internal** （因为未指定 **-h** ），而 **myNSSPassword** 的令牌密码则用于访问令牌。

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

2.5.8. CRMFPopClient

CRMFPopClient 工具是使用 NSS 数据库的证书请求消息格式(CRMF)客户端，并提供 Possession 的参与。

Example:

```
$ 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
```

这个示例在当前目录(**-d**)中创建一个带有 **cn=subject_name** 主题 DN (**-n**) 的新 CSR，用于指定用于传输 **kra.transport** (**-b**)的证书，即 **AES/CBC/PKCS5Padding** key wrap 算法详细输出(**-v**) 生成的 CSR 被写入 **/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**吊销证书”。

2.5.9. HttpClient

HttpClient 实用程序是用于提交 CMC 请求的 NSS 感知 HTTP 客户端。

Example:

```
$ 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

用于检查证书撤销状态的在线证书状态协议(OCSP)客户端。

Example:

```
$ 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** (**-p**)上查询 **server.example.com** OCSP 服务器(**-h**)，以检查由 **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。**

详情请查看 OCSPClient --help 命令的输出、更多选项和其他示例。

2.5.11. PKCS10Client

PKCS10Client 工具为 RSA 和 EC 密钥创建一个 PKCS10 格式的 CSR，可选在 HSM 上。

Example:

```
$ 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/ 目录中创建一个新的 RSA (-a)密钥，其 2048 位(-l)。输出 CSR 存储在 ~/ds.cfg 文件(-o)中，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。

Example:

```
$ 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。输出包括签名算法、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。

Example:

```
$ 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。输出包括信息，如撤销签名算法、撤销的签发者以及撤销的证书列表及其原因。

详情请查看 PrettyPrintCrl(1) man page。

2.5.14. TokenInfo

TokenInfo 实用程序列出 NSS 数据库中的所有令牌。

Example:

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

此命令列出在指定数据库目录中注册的所有令牌(HSM、软令牌等等)。

有关详情、更多选项和其他示例，请参阅 TokenInfo 命令的输出

2.5.15. tkstool

tkstool 工具与令牌密钥服务(TKS)子系统交互。

Example:

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

此命令在 HSM 令牌_name上的 /var/lib/pki/pki-tomcat/alias NSS 数据库中创建一个名为 new_master (-n)的新主密钥(-M)

详情请查看 tkstool -H 命令的输出、更多选项和其他示例。

2.6. 企业安全客户端

企业安全 客户端是 Red Hat Certificate System 的一个工具，简化了管理智能卡。最终用户可以使用安全令牌(smart 卡)来存储用于应用的用户证书，如单点登录访问和客户端身份验证。最终用户签发令牌，其中包含签名、加密和其他加密功能所需的证书和密钥。

企业安全客户端是 证书系统的完整令牌管理系统的第三部分。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。企业安全客户端是 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。企业安全客户端提供以下功能：

- 支持 JavaCard 2.1 或更高版本卡和全局平台 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 格式工厂键。

- 支持 SafeNet Smart Card 650 (SC650)。
- 注册安全令牌，以便 TPS 识别它们。
- 维护安全令牌，如使用 TPS 重新注册令牌。
- 提供有关被管理令牌或令牌的当前状态的信息。
- 支持服务器端密钥生成，以便在令牌丢失时在单独的令牌上存档并恢复密钥。

企业安全客户端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。这是证书系统令牌管理系统中的最终组件，带有令牌处理系统(TPS)和令牌密钥服务(TKS)。

企业安全客户端提供令牌管理系统的用户界面。最终用户可以发布包含签名、加密和其他加密功能所需的证书和密钥的安全令牌。要使用令牌，TPS 必须能够识别并与它们通信。企业安全客户端是要注册的令牌的方法。

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

正确注册令牌后，可将 Web 浏览器配置为识别令牌并将其用于安全操作。企业安全客户端提供以下功能：

- 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，企业安全客户端使得可以使用 TPS 重新注册令牌。
- 通过默认和自定义令牌配置集提供对多种不同类型的令牌的支持。默认情况下，TPS 可以自动注册用户密钥、设备密钥和安全密钥；可以添加额外的配置文件，以便用于不同用途的令牌（通过令牌 CUID 等属性识别）可以根据适当的配置集自动注册。

- 提供有关被管理令牌的当前状态的信息。

部分 II. 设置证书服务

第 3 章 为颁发证书（证书配置文件）进行规则

证书系统提供了一个可自定义的框架，用于应用传入证书请求的策略，并控制输入请求类型和输出证书类型；它们称为 *证书配置文件*。证书配置文件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文件。

3.1. 关于证书配置文件

证书配置文件定义了与发布特定类型的证书相关的所有内容，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。注册和续订请求被提交到证书配置文件，然后受到该证书配置文件中设置的默认值和约束。这些限制是否通过与证书配置文件关联的输入表单提交，还是通过其他方式提交。从证书配置文件请求发布的证书包含默认值所需的内容以及默认参数所需的信息。约束提供证书中允许的内容的规则。

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

证书系统包含一组默认配置文件。虽然创建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文件或修改现有的配置文件。

- *身份验证*。每个认证配置文件中都可以指定身份验证方法。
- *授权*。每个认证配置文件中都可以指定一个授权方法。
- *配置集输入*。配置集输入是请求证书时提交到 CA 的参数和值。配置集输入包括证书请求的公钥，以及证书的最终实体所请求的证书主题名称。
- *配置集输出*。配置集输出是参数和值，用于指定向最终实体提供证书的格式。当请求成功时，配置集输出是 CMC 响应，其中包含 PKCS the7 证书链。
- *证书内容*。每个证书定义内容信息，如为其分配的实体名称（主题名称）、其签名算法及其有效期周期。证书中包含的内容在 X.509 标准中定义。使用 X509 标准的版本 3 时，证书也可以包含扩展。有关证书扩展的详情，请参考 ???。

有关证书配置文件的所有信息都在配置文件配置文件中配置文件策略的 设置 条目中定义。当同时请求多个证书时，可以在配置文件策略中定义多个设置条目来满足每个证书的需求。每个策

略集由多个策略规则组成，每个策略规则描述了证书内容中的一个字段。策略规则可包括以下部分：

- *配置集默认值。*这些是预定义的参数和允许的值，用于证书中包含的信息。配置集默认包括证书的有效性周期，以及每个发布的证书类型的证书扩展。

- *配置集限制。*用于发布证书的约束设置规则或策略。另一方面，配置集限制包括要求证书主题名称至少有一个 CN 组件的规则，将证书的有效性设置为最多 360 天，来定义续订允许的宽限期，或要求 主题altname 扩展始终设为 true。

3.1.1. Enrollment Profile

表 11.1 中更详细地列出了定义输入、输出和策略集的每个配置集的参数。Red Hat Certificate System Planning, Installation and Deployment Guide 中的 profile 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

证书配置文件的第一个部分是描述。这将显示名称、长描述、是否启用它以及是否启用了谁。

```
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

此配置集中缺少 auth.instance_id= 条目意味着使用此配置集，不需要进行身份验证来提交注册请求。但是，需要由授权的 CA 代理手动批准才能获得颁发。

接下来，配置集列出了配置集所需的所有输入：

```
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，这意味着最终证书的格式。唯一可用的是 certOutputImpl，这会导致 CMC 响应返回给请求者（如果成功）。

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

最后 - 最大配置块是为配置集设置的策略。Policy set 列出应用于最终证书的所有设置，如其有效期、其续订设置以及证书可用于的操作。policyset.list 参数标识应用到一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，第六个策略根据策略中的配置，在证书中自动填充密钥用法扩展。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：默认和约束

扩展配置附加信息，以包含在证书或规则中有关如何使用证书。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，然后根据约束强制执行。

如果请求中没有设置证书扩展，则通过添加 *默认值*，在配置集中添加或识别证书扩展。例如，基本限制扩展标识证书是否为 CA 签名证书、可以在 CA 下配置的最大从属 CA 数，以及扩展是否至关重要（必需的）：

```
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
```

此外，扩展也可以为证书请求设置所需的值，名为 *约束*。如果请求的内容与集合约束不匹配，则拒绝请求。约束通常与扩展默认对应，但并不总是始终对应。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

要允许用户提供的扩展嵌入到证书请求中，并忽略配置集中的系统定义默认值，配置集需要包含 User Supplied Extension Default，如 第 B.1.32 节 “用户 Supplied 扩展默认” 所述。

3.1.3. 输入和输出

输入设置必须提交才能接收证书的信息。这可以是请求者信息、特定格式的证书请求或组织信息。

配置集中配置的输出定义发布的证书的格式。

在证书系统中，通过 *注册表单* 访问配置文件，该表单可通过终端实体页面访问。（即使客户端（如 TPS ）通过这些表单提交注册请求。） 输入然后对应于注册表单中的字段。输出对应于证书检索页面中包含的信息。

3.2. 设置证书配置文件

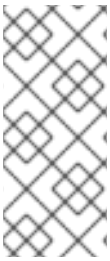
在证书系统中，您可以添加、删除和修改注册配置文件：

- 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文件

本节描述了如何使用 pki 工具管理证书配置文件。详情请查看 pki-ca-profile(1) man page。

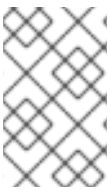


注意

建议使用 raw 格式。有关配置文件的每个属性和字段的详情，请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide 中的在文件系统中直接创建和编辑证书配置文件一节。

3.2.1.1. 启用和禁用证书配置文件

在编辑证书配置文件前，您必须禁用它。修改完成后，您可以重新启用配置集。



注意

只有 CA 代理才能启用和禁用证书配置文件。

例如，禁用 caCMCECserverCert 证书配置文件：

```
-
```

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

例如，启用 caCMCECserverCert 证书配置文件：

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

3.2.1.2. 以 Raw 格式创建证书配置文件

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，按如下方式指定新配置集 ID：

```
profileId=profile_name
```

3.2.1.3. 以 Raw 格式编辑证书配置文件

CA 管理员可以以原始格式编辑证书配置文件，而无需手动下载配置文件。

例如，编辑 caCMCECserverCert 配置集：

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

此命令以 raw 格式自动下载配置文件配置，并在 VI 编辑器中打开它。当您关闭编辑器时，配置集配置会在服务器上更新。

编辑配置集后您不需要重启 CA。

重要

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

-

例 3.2. 以 RAW 格式编辑证书配置文件

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 ???。

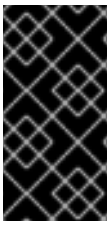
3. 启用配置集作为 CA 代理：

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

3.2.1.4. 删除证书配置文件

删除证书配置文件：

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



重要

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

3.2.2. 使用基于 Java 的管理控制台管理证书注册配置文件

重要

pkiconsole 已被弃用。

3.2.2.1. 通过 CA 控制台创建证书配置文件

为安全起见，证书系统会强制分离现有证书配置文件，只要代理允许后管理员才能对其进行编辑。要添加新证书配置文件或修改现有证书配置文件，请以管理员身份执行以下步骤：

1. 登录证书系统 CA 子系统控制台。

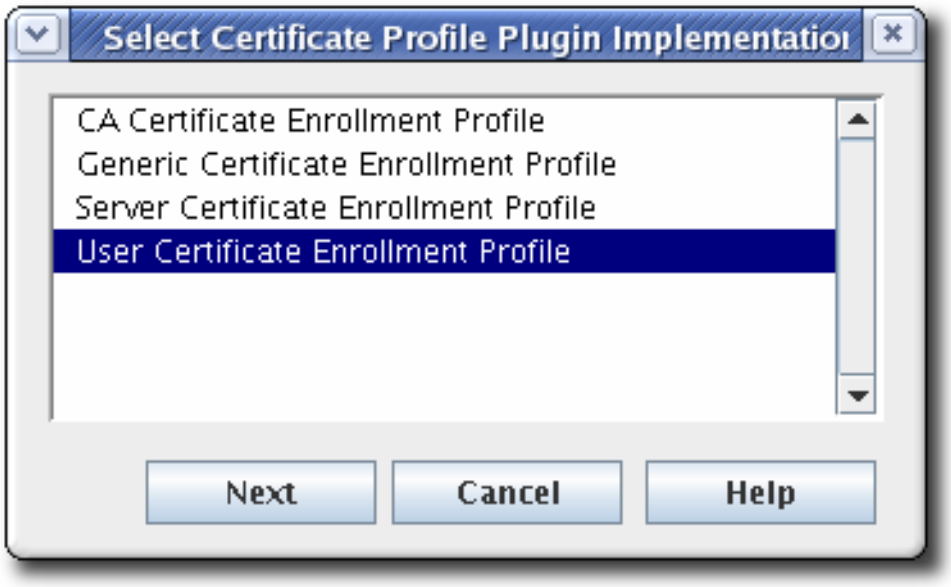
```
pkiconsole https://server.example.com:8443/ca
```

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

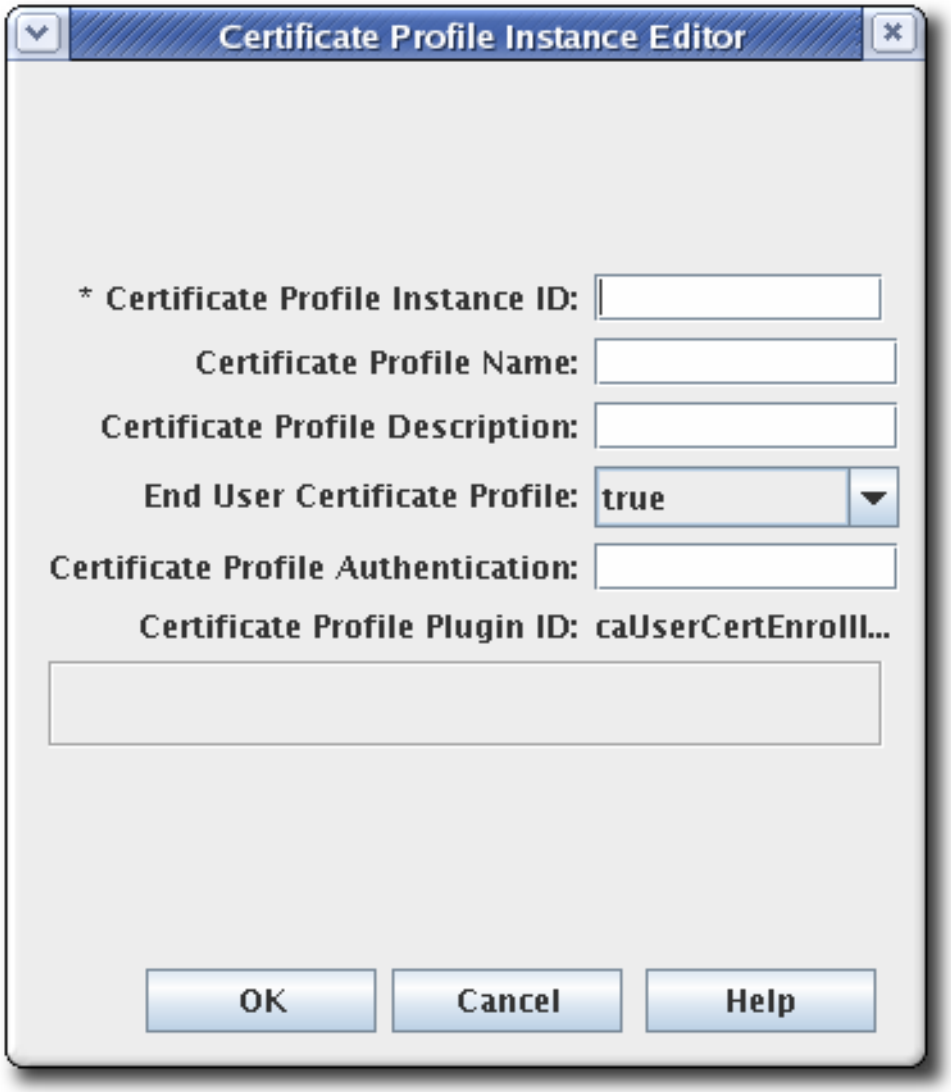
   Certificate Profile Instances Management 选项卡（列出配置的证书配置文件）将打开。

3. 若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 中填写配置文件信息。



- 证书配置文件实例 ID.这是系统用来识别配置文件的 ID。
- 证书配置文件名称.这是配置文件的用户友好的名称。
- 证书配置文件描述.
- 最终用户证书配置文件.这会设置请求是否必须通过配置文件的输入表单进行。这通常设为 true。把它设置为 false，允许通过证书管理器的证书配置文件框架处理签名的请求，而不是通过证书配置文件的输入页面进行处理。
- 证书配置文件身份验证.这会设置身份验证方法。通过为身份验证实例提供实例 ID 来设

置自动身份验证。如果此字段为空，则身份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中。

除非是 TMS 子系统，否则管理员必须选择以下身份验证插件之一：

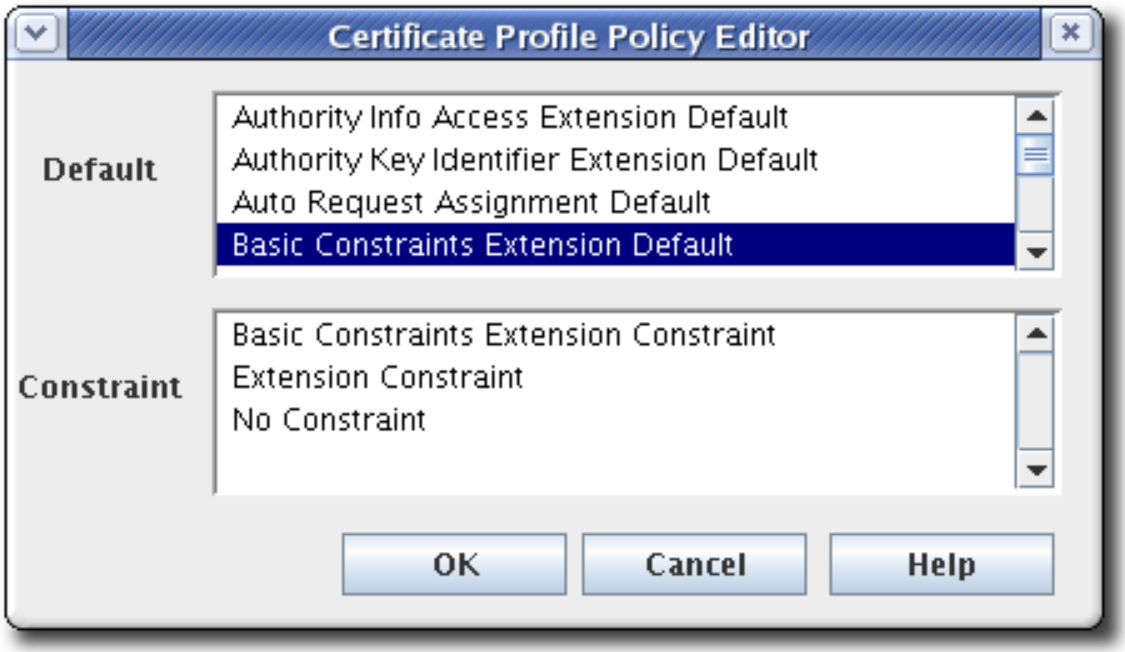
- CMCAuth ：当 CA 代理必须批准并提交注册请求时，使用此插件。
- CMCUserSignedAuth ：使用此插件使非代理用户能够注册自己的证书。

5. 点确定。插件编辑器关闭，新的配置集列在 profile 选项卡中。

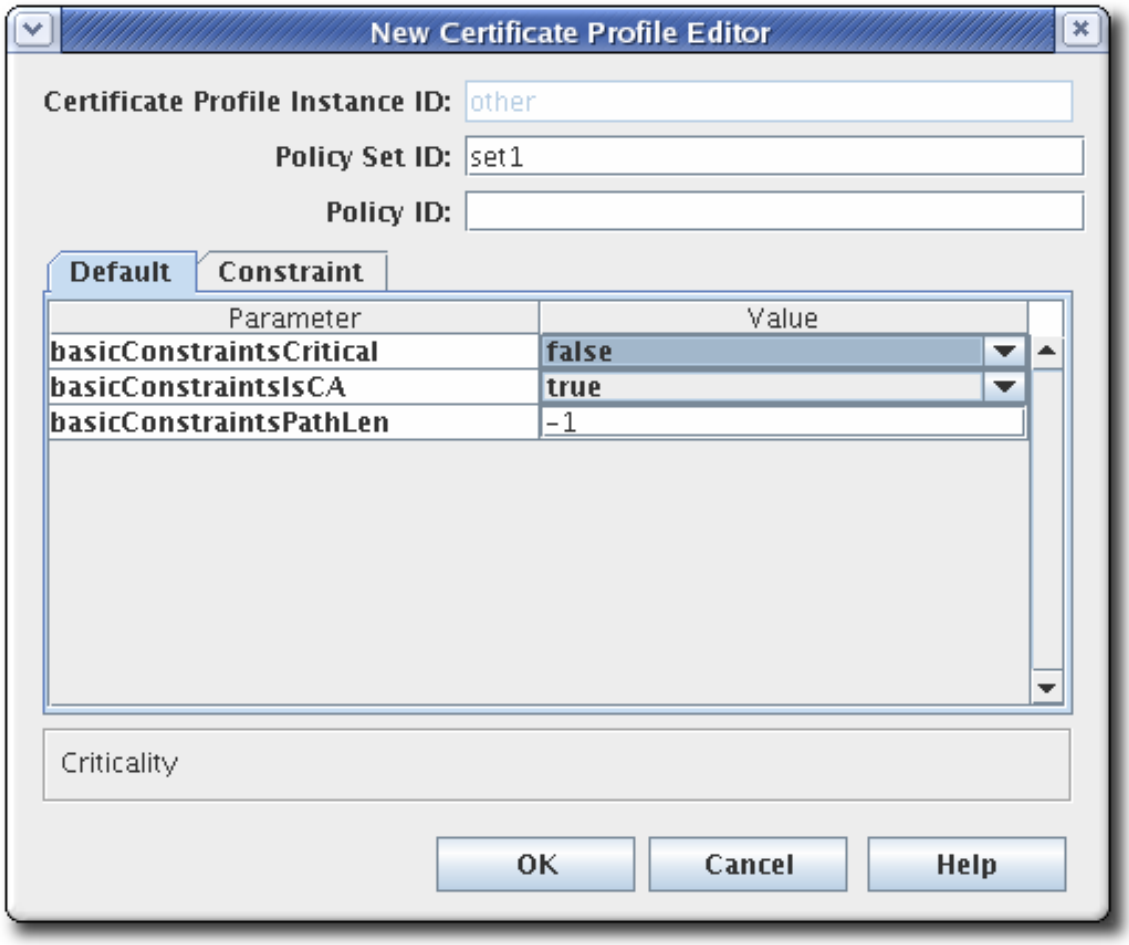
6. 为新配置集配置策略、输入和输出。从列表中选择新配置集，然后单击 Edit/View。

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 选项卡列出了已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约束，然后单击 OK。

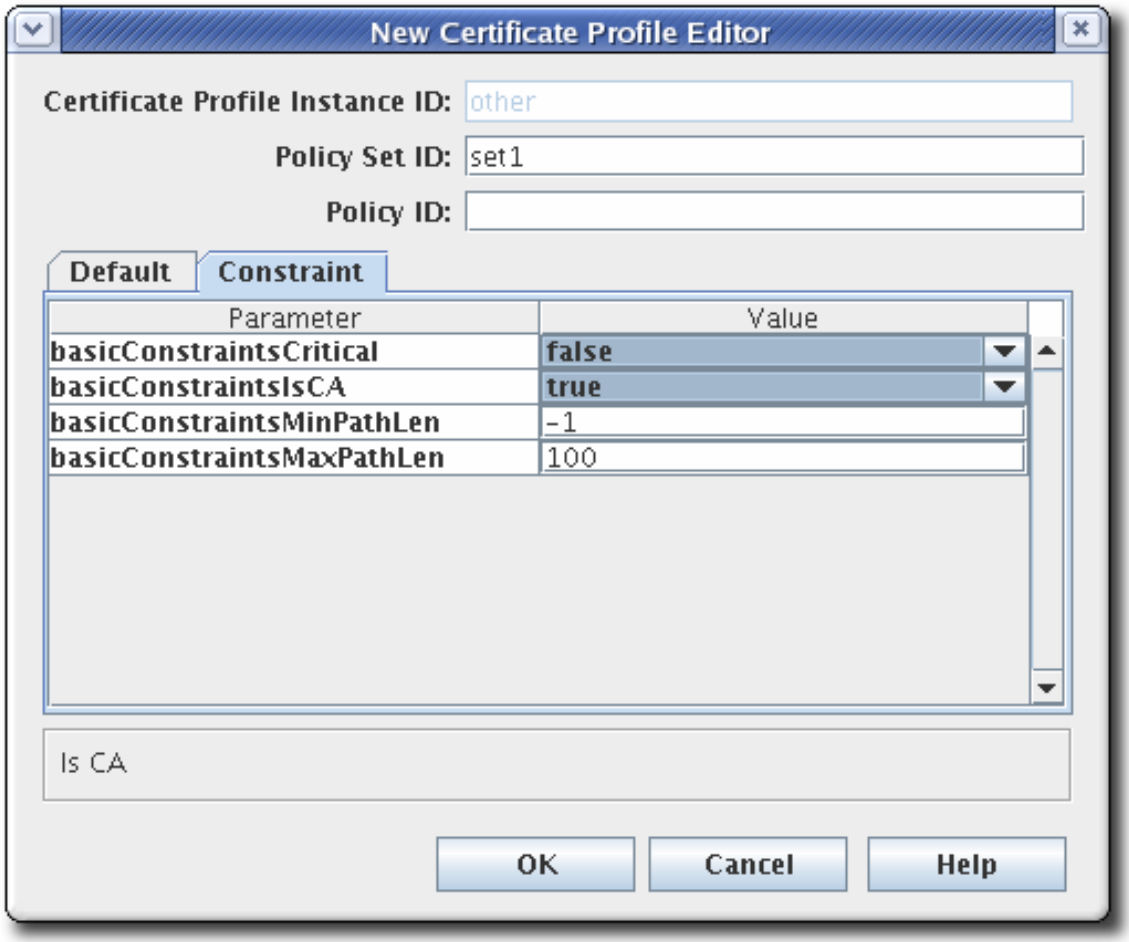


c.

填写策略设置 ID。在发布双密钥对时，单独的策略集会定义与每个证书关联的策略。然后，填写证书配置文件策略 ID，证书配置文件策略的名称或标识符。

d.

在 Defaults 和 Constraints 选项卡中配置任何参数。



Default 定义填充证书请求的属性，后者决定证书的内容。这些可以是扩展、有效期或证书中包含的其他字段。约束 定义默认值的有效值。

如需了解每个默认或约束的详情，请查看 第 B.1 节 “默认参考” 和 第 B.2 节 “约束参考”。

要修改现有策略，请选择策略，然后单击 **Edit**。然后编辑该策略的默认和限制。

要删除策略，请选择策略，然后单击 **Delete**。

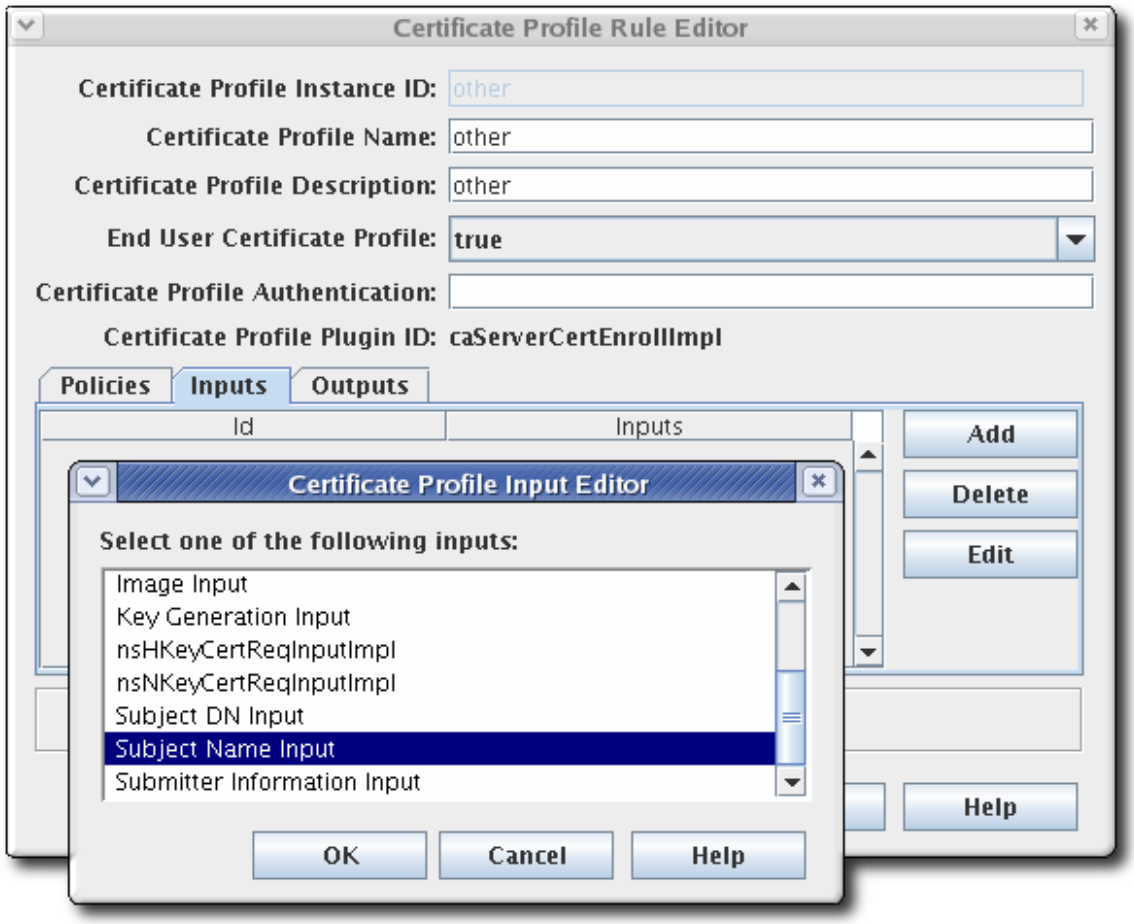
8. 在证书配置文件 编辑器 窗口的 **Inputs** 选项卡中设置输入。配置集可以有多个输入类型。

注意

除非为 **TMS** 子系统配置配置文件，否则请选择 **cmcCertReqInput**，并通过选择它们并单击 **Delete** 按钮来删除其他配置集。

a.

要添加输入，请点击 Add。

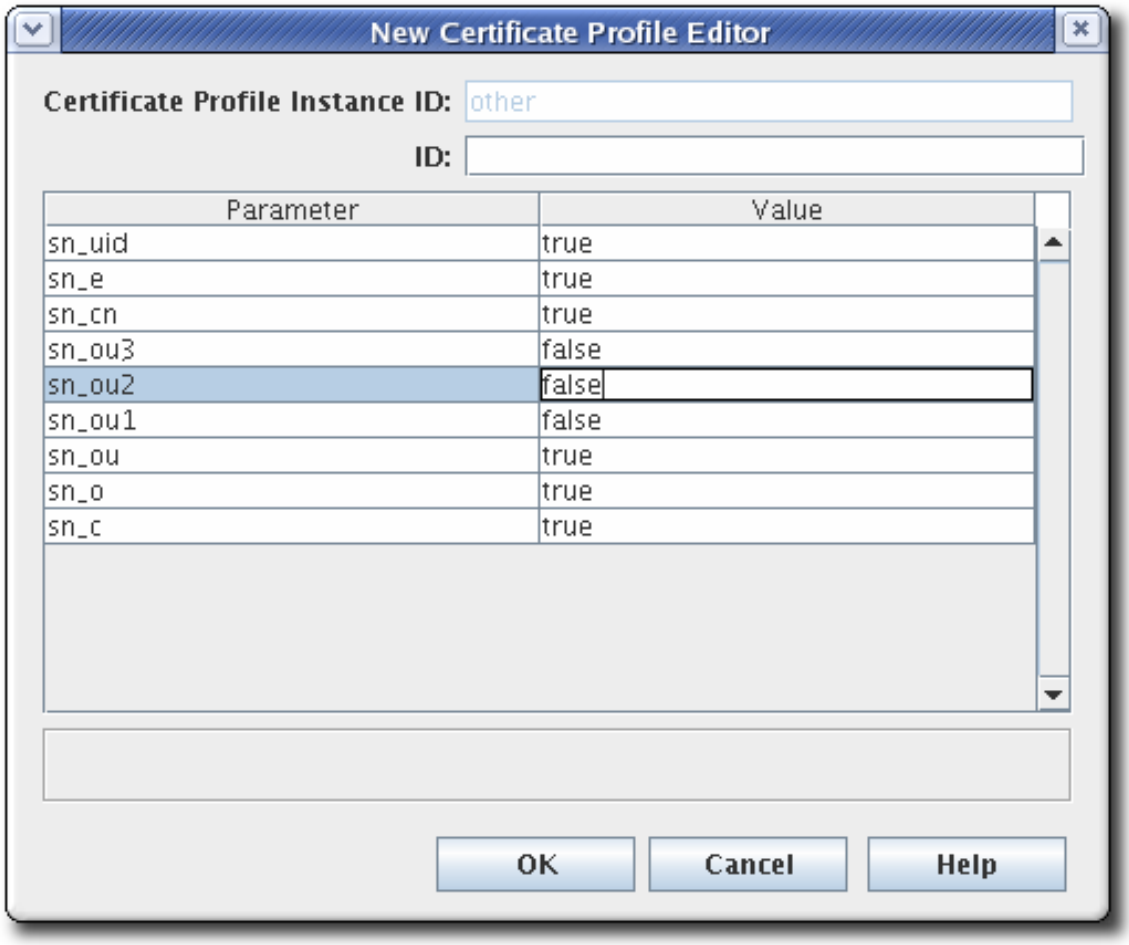


b.

从列表中选择输入，然后单击确定。如需了解默认输入的详情，请查看 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 OK。



可以添加和删除输入。可以选择为输入选择编辑，但因为输入没有参数或其他设置，因此无需配置。

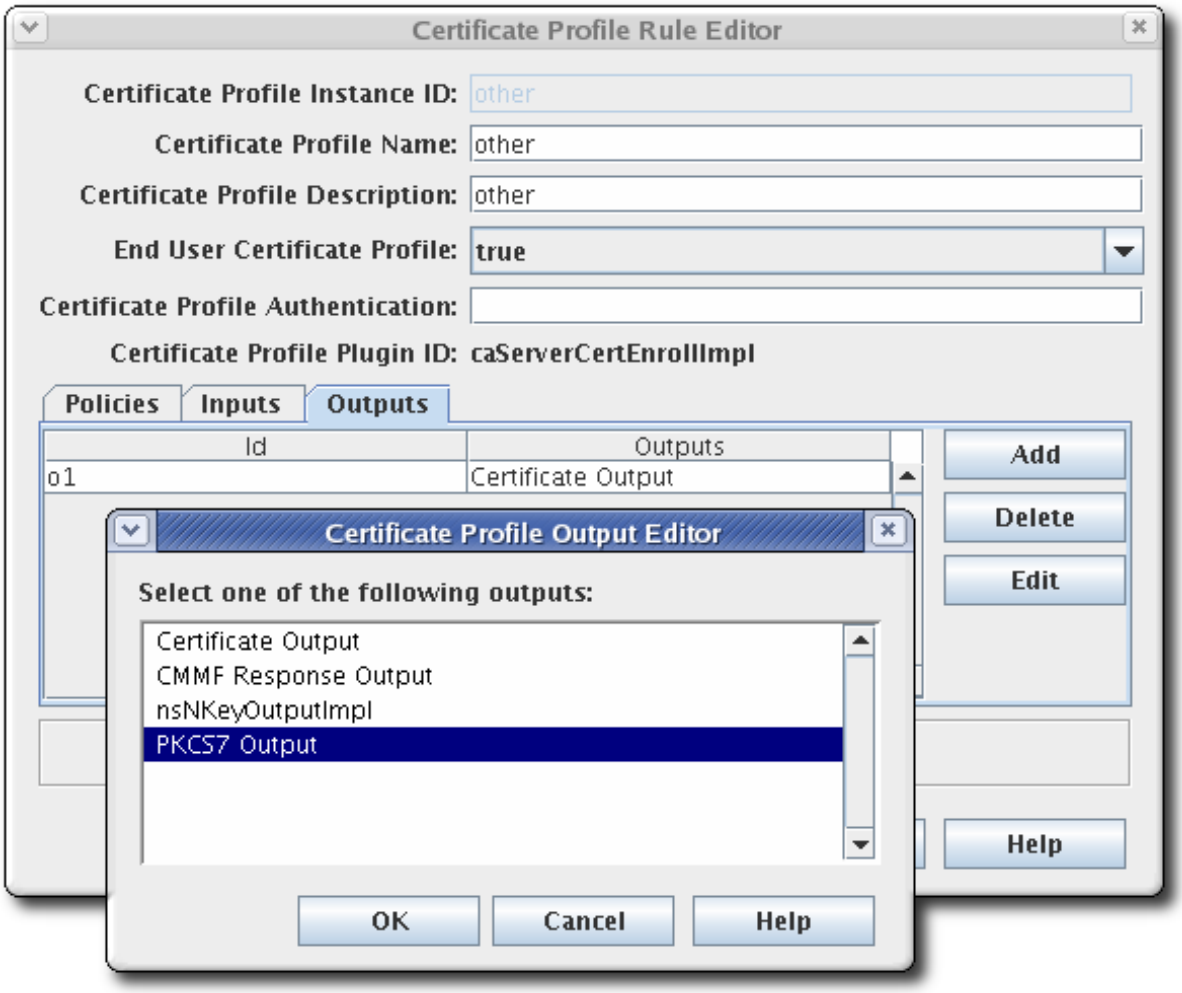
要删除输入，请选择输入，然后单击 **Delete**。

9.

在证书配置文件 规则编辑器 窗口的 **Outputs** 选项卡中设置 输出。

必须为使用自动验证方法的任何证书配置文件设置输出；不需要为使用代理验证的任何证书配置文件设置输出。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中。

除非为 **TMS** 子系统配置配置文件，否则仅选择证书 输出。



可以添加和删除输出。可以为输出选择编辑，但因为输出没有参数或其他设置，因此无需配置。

a. 要添加输出，请单击 **Add**。

b. 从列表中选择输出，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，以便为此输出中的参数提供值。

若要删除输出，可从列表中选择输出，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. 将配置集作为管理员创建后，CA 代理必须批准代理服务页面中的配置集来启用配置集。

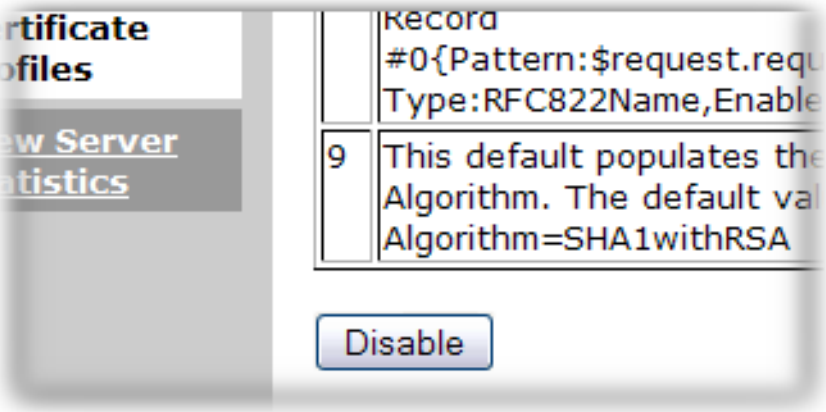
a. 打开 CA 的服务页面。

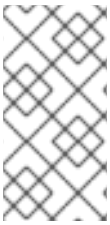
```
https://server.example.com:8443/ca/services
```

b. 单击 Manage Certificate Profiles 链接。本页列出了管理员（活跃和不活跃）设置的所有证书配置文件。

c. 单击要批准的证书配置文件的名称。

d. 在页面底部，单击 Enable 按钮。





注意

如果此配置文件与 TPS 一起使用，则必须配置 TPS 来识别配置文件类型。这是 11.1.4 中。红帽证书系统规划、安装和部署指南中的管理智能卡 CA 配置文件。

配置文件的授权方法只能使用命令行添加到配置集，如 Red Hat Certificate System Planning, Installation and Deployment Guide 中的直接创建和编辑证书配置文件一节中所述。

3.2.2.2. 在控制台中编辑证书配置文件

修改现有证书配置文件：

1. 登录到代理服务页面并禁用配置集。

   代理启用证书配置文件后，该证书配置文件会在证书配置文件 实例管理 选项卡中被标记为启用，并且无法通过控制台以任何方式编辑证书配置文件。

2. 登录证书系统 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，然后单击 Edit/View。

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

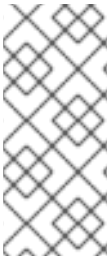
   注意

   无法修改配置集实例 ID。

   如有必要，通过拔出窗口其中一个角来放大窗口。

6. 重启 CA 以应用更改。

7. 在代理服务页面中，重新启用配置集。



注意

删除未被代理批准的任何证书配置文件。证书配置文件实例管理选项卡 中出现的任何证书配置文件 也会出现在代理服务界面中。如果已经启用配置集，则必须禁用代理，然后才能从配置集列表中删除它。

3.2.3. 列出证书注册配置文件

安装证书系统 CA 时，以下预定义的证书配置文件可以使用并在此环境中设置。这些证书配置文件是为最常见的证书类型设计的，它们提供了常见的默认值、约束、身份验证方法、输入和输出。

若要列出命令行中的可用配置文件，请使用 pki 实用程序。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。如需了解更多信息，请参阅 *红帽认证系统规划、安装和部署指南*。

3.2.4. 显示证书注册配置文件的详情

例如，显示特定的证书配置文件，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文件，如 caECFullCMCUserSignedCert，格式为 raw：

```
$ 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

在创建证书配置文件时，*必须在 Subject Key Identifier Default 前添加 Key Default*。证书系统在创建或应用 Subject Key Identifier Default 之前处理 Key Default 中的密钥约束，因此如果密钥尚未处理，在主题名称中设置密钥会失败。

例如，object-signing 配置集可能会定义这两个默认值：

```
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)前列出密钥默认(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文件。有关如何续订证书的详情，请参考 第 5.4 节 “续订证书”。

允许续订的配置集通常由 renewal GracePeriodConstraint 条目组成。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. 使用相同密钥续订

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，请使用带有新密钥的相同配置文件。证书系统使用来自用于为新证书发出请求的用户签名的证书的 subjectDN。

3.5. 为证书设置签名算法

CA 的签名证书可以签署它通过 CA 支持的任何公钥算法发布的证书。例如，ECC 签名证书可以同时为 ECC 和 RSA 证书请求签名，只要 CA 支持 ECC 和 RSA 算法。RSA 签名证书可以使用 EC 密钥为 PKCS11410 请求签名，但如果 ECC 模块没有可用于 CA 验证 CRMF 概念验证(POP)，则可能无法使用 ECC 模块为 CRMF 证书签名。

ECC 和 RSA 是公钥加密和解密算法。公钥算法都支持不同的密码套件，用于加密和解密数据的算法。CA 签名证书的功能部分就是使用其其中一个支持的密码套件发布和签署证书。

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，则配置集会使用任何默认签名算法。

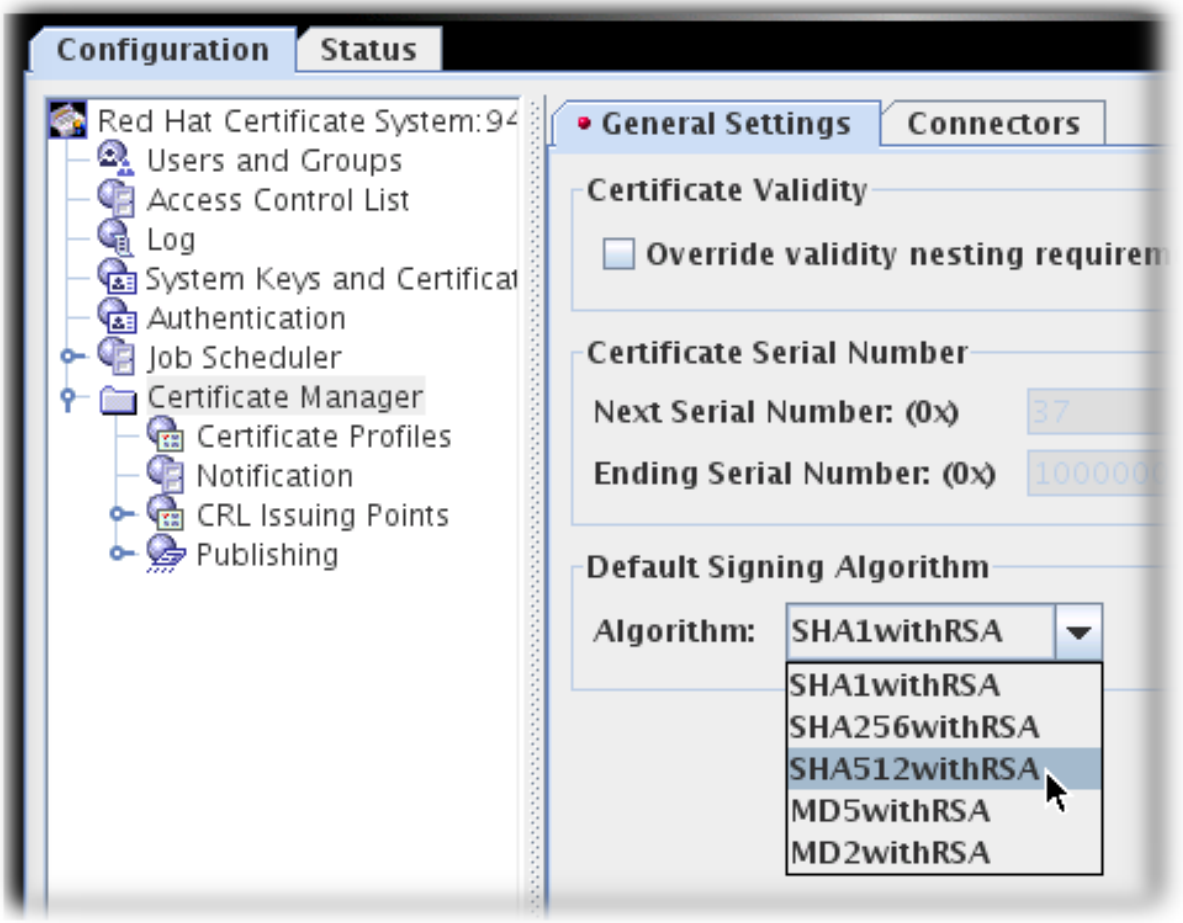
3.5.1. 设置 CA 的默认签名算法

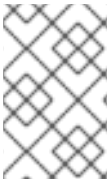
1. 打开 CA 控制台。

```
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，在 Algorithm 下拉菜单中选择要使用的算法。





注意

pkiconsole 已被弃用。

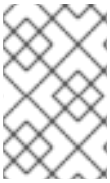
3.5.2. 在配置文件中设置签名算法默认值

每个配置集都定义了 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，则配置集将使用任何设置为 CA 的默认值。

在配置文件的 .cfg 文件中，算法使用两个参数设置：

```
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，必须首先由代理禁用它。

1. 打开 CA 控制台。

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

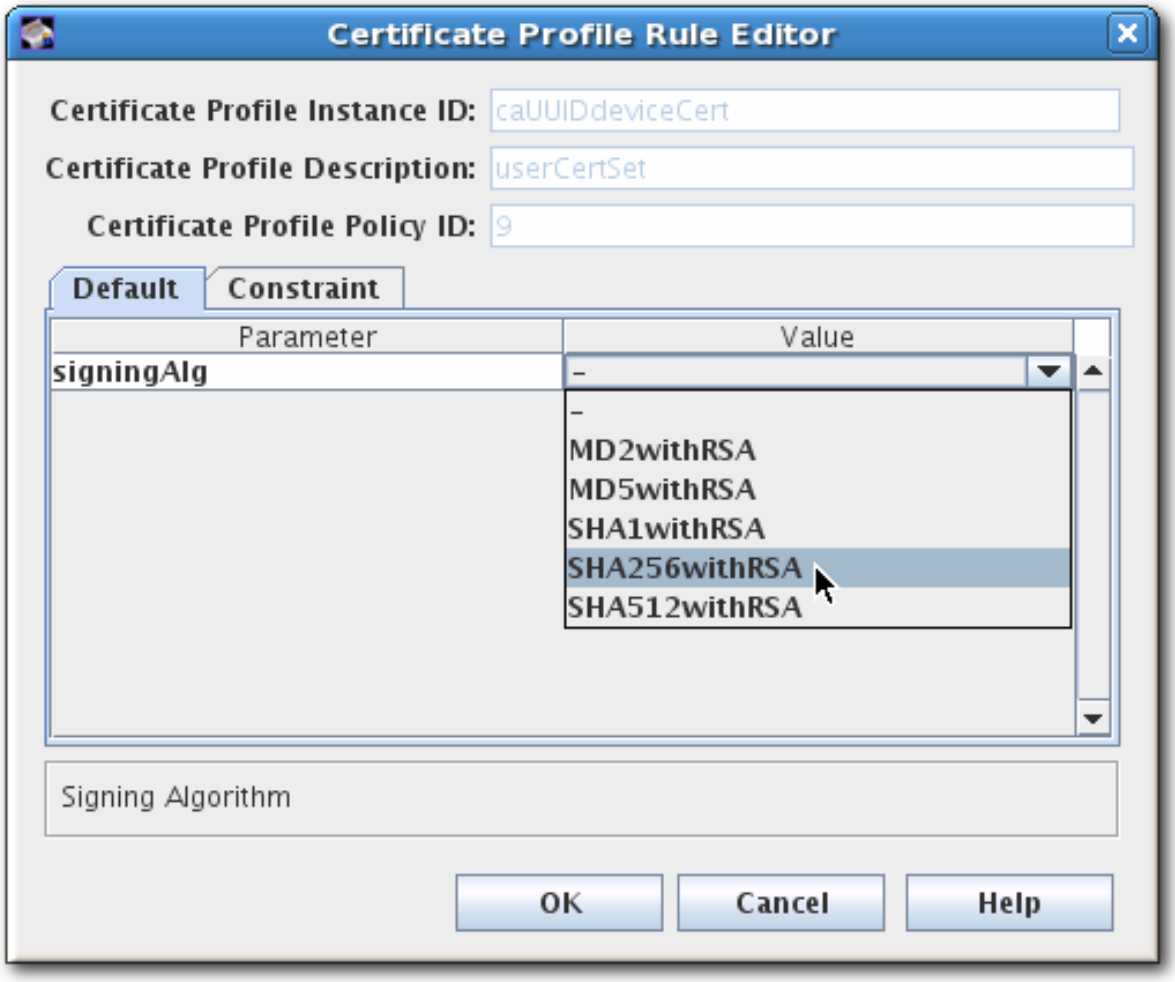
2. 在 Configuration 选项卡中，展开 证书管理器 树。

3. 点 Certificate Profiles 项。

4. 点 Policies 选项卡。

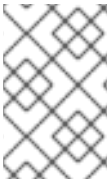
5. 选择 Signing Alg 策略，然后单击 Edit 按钮。

6. 要设置默认签名算法，请在 Defaults 选项卡中设置值。如果设置为 -，则配置文件将使用 CA 的默认值。



7. 要设置可在证书请求中可接受的允许签名算法列表，请打开 Constraints 选项卡，并在 Value 字段中为 signingAlgsAllowed 设置算法列表。

   约束的可能值列在 第 B.2.10 节 “签名算法约束” 中。



注意

pkiconsole 已被弃用。

3.6. 管理 CARELATED 配置集

必须使用证书配置文件和扩展来设置从属 CA 如何发布证书的规则。这有两个部分：

- 管理 CA 签名证书

- 定义颁发规则

3.6.1. 在 CA 证书中设置限制

创建从属 CA 时，根 CA 可能会对从属 CA 实施限制或限制。例如，root CA 可以指定有效认证路径的最大深度（允许通过设置 CA 签名证书中基本限制扩展的子 CA 数量）。

证书链通常由实体证书、零个或多个中间 CA 证书以及 root CA 证书组成。root CA 证书是自签名的，也可以是由外部可信 CA 签名。发布后，root CA 证书作为可信 CA 加载到证书数据库中。

在执行 TLS 握手、发送 S/MIME 消息或发送已签名对象时，证书交换将发生。作为握手的一部分，发送方应该发送主题证书以及将主题证书链接到可信 root 所需的任何中间 CA 证书。要使证书链正常工作，证书应该具有以下属性：

- CA 证书必须具有基本约束扩展。
- CA 证书必须在 Key Usage 扩展中设置 keyCertSign 位。
- 当 CA 生成新密钥时，它们必须为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。CA 证书必须包含 Subject Key Identifier 扩展。

有关证书及其扩展的更多信息，请参阅 *Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile (RFC 539)*），请参阅 RFC 5280。

这些扩展可以通过证书配置文件注册页面进行配置。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

此流程描述了编辑 CA 用来向其从属 CA 发布 CA 证书的 CA 证书配置文件。

CA 实例首次配置时使用的配置集是 /var/lib/pki/*instance_name*/ca/conf/caCert.profile。在 pkiconsole 中无法编辑此配置集（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值：

1. 如果当前启用了配置集，则必须禁用它，然后才能编辑它。打开 agent services 页面，从左侧导航菜单中选择 Manage Certificate Profiles，选择 profile，然后单击 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 证书管理器，然后选择 证书配置文件。

4. 从右窗口中，选择 caCACert 或适当的 CA 签名证书配置文件，然后单击 Edit/View。

5. 在 Certificate Profile Rule Editor 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 Constraint （根据默认值）。

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

8. 为 CA 证书设置约束值。没有为 Key Usage 扩展设置限制；对于扩展的密钥用法扩展，请为 CA 设置适当的 OID 约束。如需更多信息，请参阅 第 B.1.8 节 “扩展密钥用法扩展默认值”。

9. 当对配置文件进行更改时，再次登录到代理服务页面，然后重新启用证书配置文件。

注意

pkiconsole 已被弃用。

有关修改证书配置文件的详情，请参考 第 3.2 节 “设置证书配置文件”。

3.6.2. 在颁发证书上更改 CA 的限制

在配置了子系统后，会默认设置对发布的证书的限制。它们是：

- 证书是否可以用有效期超过 CA 签名证书发布。默认值为禁止此操作。
- 用于签署证书的签名算法。
- CA 可用于发布证书的序列号范围。

从属 CA 对有效期、证书类型以及他们可以发布的扩展类型具有限制。从属 CA 可能会发布违反这些限制的证书，但验证违反这些限制的证书的客户端将不接受该证书。在更改从属 CA 的发布规则前，检查 CA 签名证书上设置的限制。

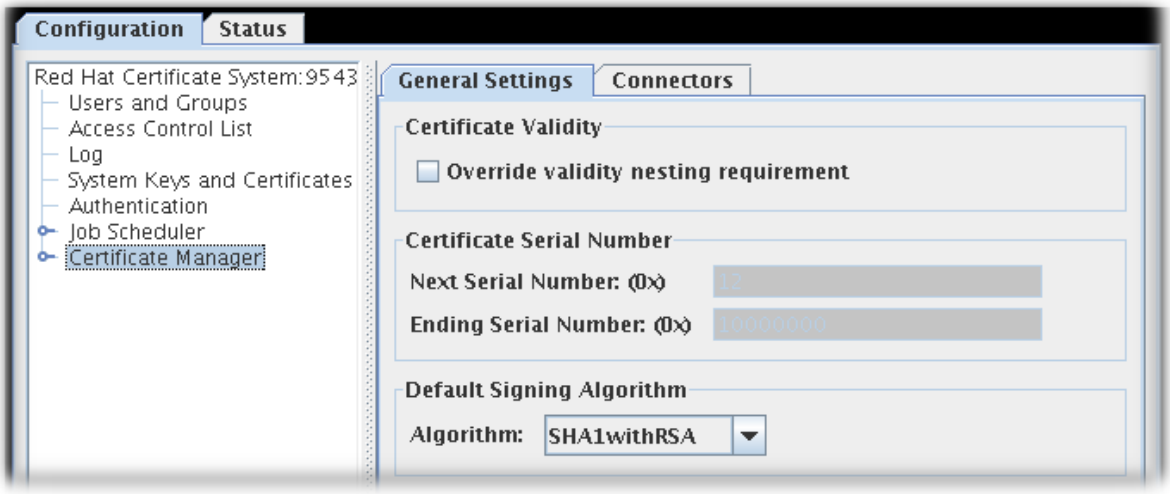
更改证书颁发规则：

1. 打开证书系统控制台。

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

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

图 3.1. 默认情况下，非从属 CA 中的 General Settings Tab



3.

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

- 覆盖有效期嵌套要求。此复选框设定证书管理器是否可以发布有效期超过 CA 签名证书有效期的证书。

  如果没有选择此复选框，并且 CA 收到有效期超过 CA 签名证书的有效性周期的请求，它会自动截断 CA 签名证书过期之日的有效周期。

- 证书序列号。这些字段显示证书管理器发布的证书的序列号范围。服务器在 Next serial number 字段中为它发布的下一个证书分配序列号，并将 结束序列号中的数字 分配给它发布的最后一个证书。

  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唯一标识证书。

注意

克隆的 CA 的序列号范围是 fluid。所有克隆的 CA 共享一个通用配置条目，该条目定义下一个可用范围。当一个 CA 开始在可用数量中运行时，它会检查此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 获得新范围。

范围在 *begin047Number* 和 *end the the theNumber* 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
```

可以为未克隆的 CA 启用序列号管理。 但是，默认情况下，序列号管理被禁用，除非自动启用系统克隆系统。

无法通过控制台手动更新序列号范围。序列号范围是只读字段。

- 默认符号算法.指定证书管理器用来签署证书的签名算法。如果 CA 的签名密钥类型是 RSA，则选项为 SHA256withRSA 和 SHA512 withRSA。

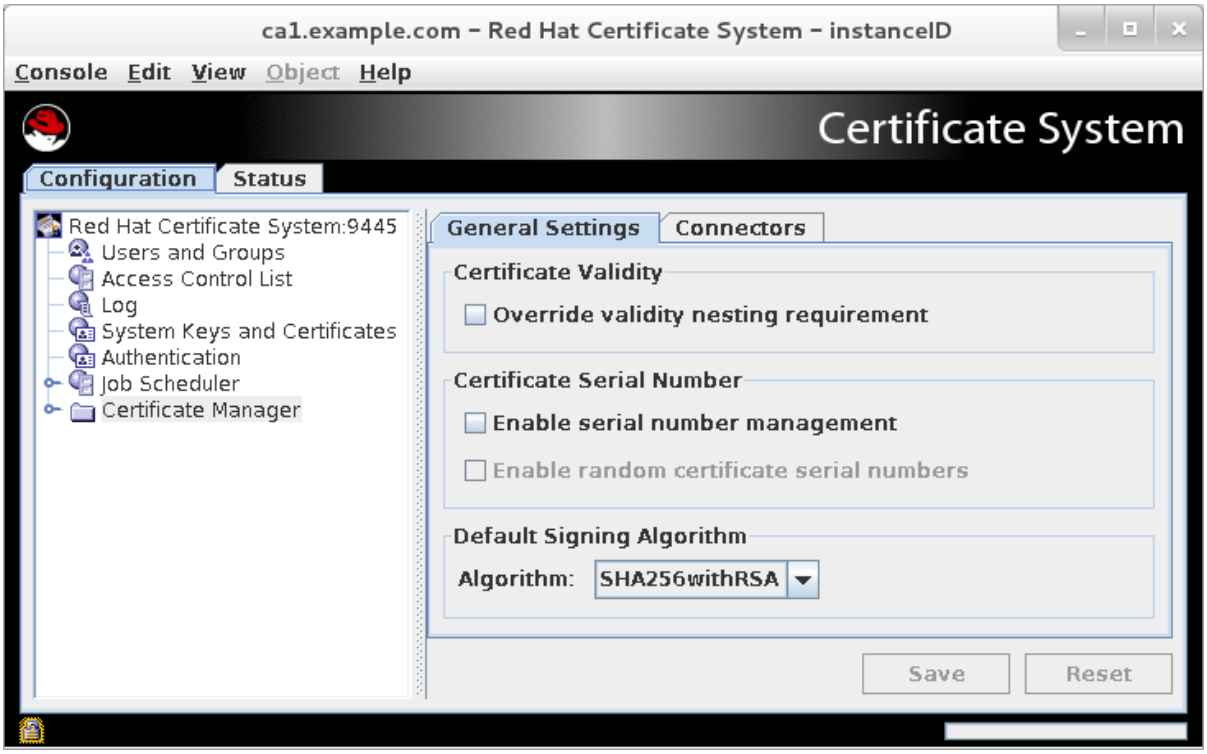
  证书配置文件配置中指定的签名算法覆盖此处设置的算法。

4. 默认情况下，在克隆的 CA 中，证书管理器 菜单项的 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 Tab



5. 点击 **Save**。

注意

pkiconsole 已被弃用。

3.6.3. 使用随机证书序列号

Red Hat Certificate System 包含用于请求、证书和副本 ID 的序列号范围管理。这允许在 安装身份管理 (IdM)时自动执行克隆。

可以通过多种方式降低基于哈希的攻击的可能性：

- 对攻击者造成证书序列号的一部分无法预计
- 在身份中添加随机选择的组件

- 通过将每个转发或后向向攻击者造成有效日期无法预测

随机证书序列号分配 方法向身份添加一个随机选择的组件。这个方法：

- 可以与克隆一起工作
- 允许解决冲突
- 与当前的序列号管理方法兼容
- 与管理员、代理和结束实体的当前工作流兼容
- 修复后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。

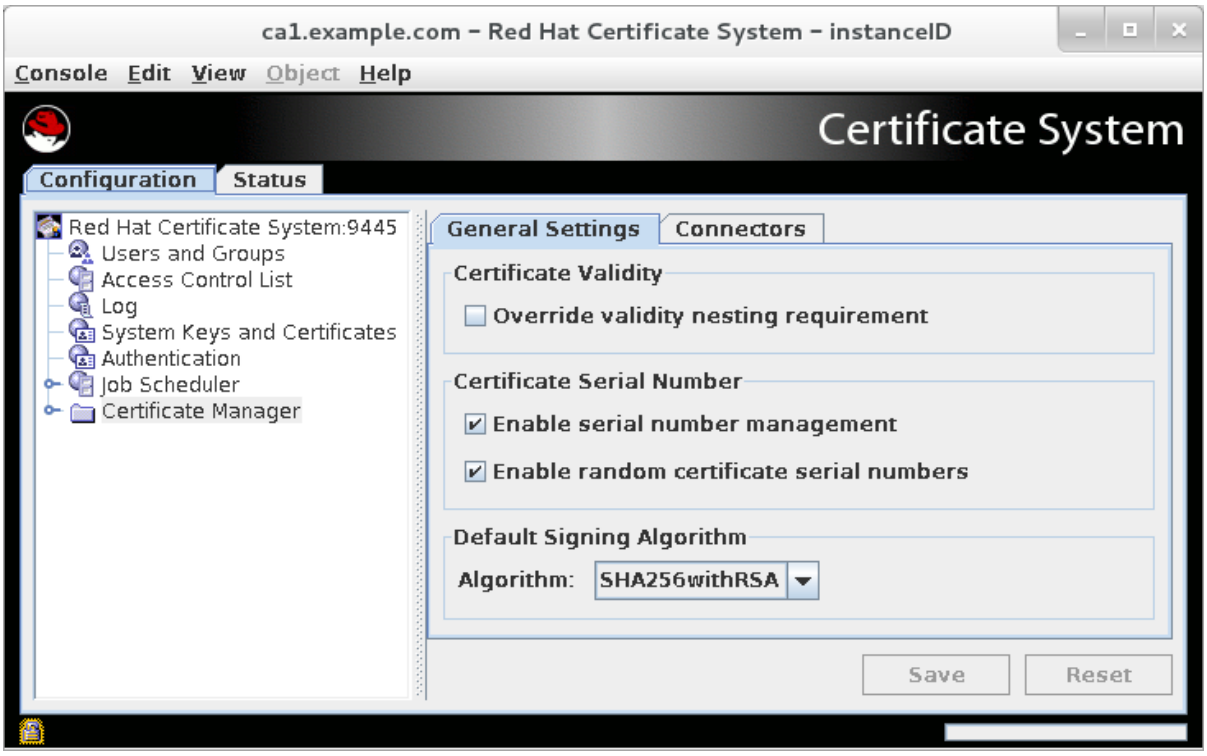
3.6.3.1. 启用随机证书颁发机构号

您可以从命令行或控制台 UI 启用自动序列号范围管理。

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 management 选项。

图 3.3. 启用随机序列号分配时常规设置标签



2. 选择 **Enable random certificate serial number** 选项。

注意

pkiconsole 已被弃用。

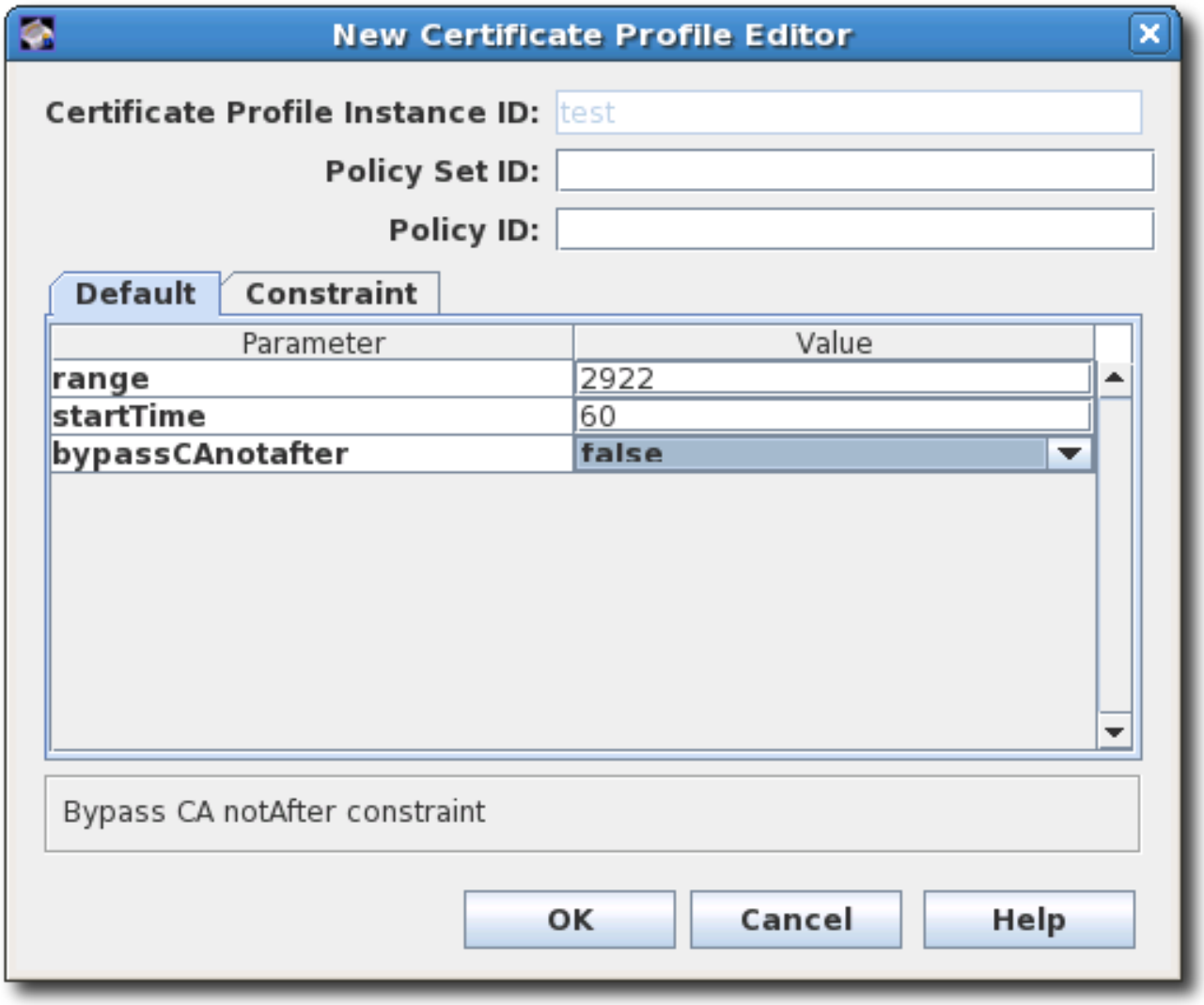
3.6.4. 允许 CA 证书重新续订 CA 的 Validity Period

通常，证书无法发布 CA 证书的过期日期 *后* 过期的时间。如果 CA 证书的过期日期为 2015 年 12 月 31 日，那么它发出的所有证书都必须由 2015 年 12 月 31 日过期。

此规则适用于 CA 发布的其他 CA 签名证书 - 这会使续订 root CA 证书几乎不可能。续订 CA 签名证书意味着它必须有有效期超过其自身的过期日期。

可以使用 CA Validity Default 更改此行为。此默认允许设置(bypassCAnotafter)，它允许使用超过签发的 CA 过期(*notAfter*)日期的有效周期发布 CA 证书。

图 3.4. CA 验证的默认配置



在实际部署中，这意味着当可能阻止 root CA 的 CA 证书时，可以续订它。

启用 CA 证书续订过去原始 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 Validity 默认。将值设为 true 以允许续订 CA 证书超过签发 CA 的有效性周期。

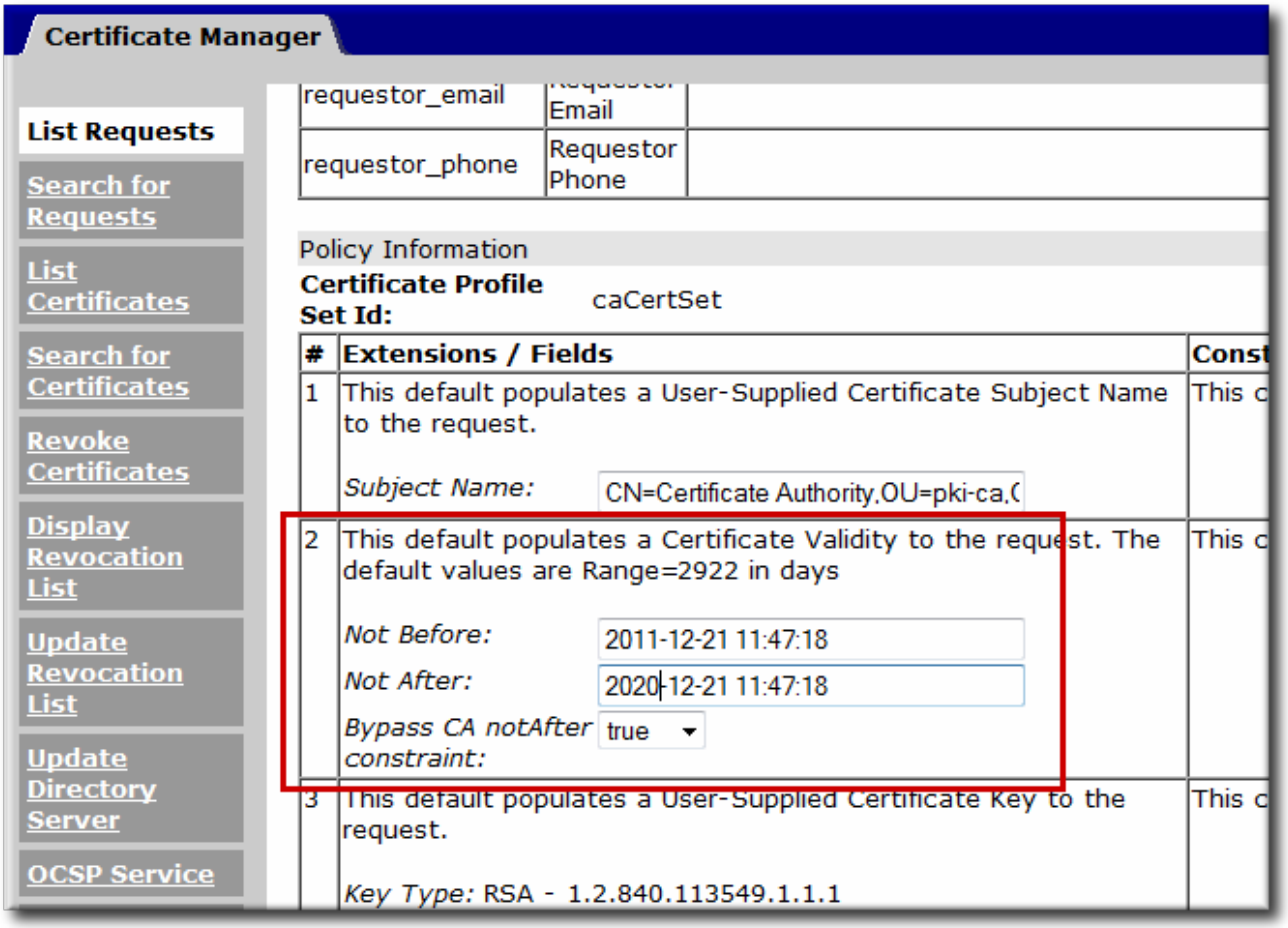
```
policyset.caCertSet.2.default.name=CA Certificate Validity Default
policyset.caCertSet.2.default.params.range=2922
policyset.caCertSet.2.default.params.startTime=0
```

```
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，允许代理绕过正常的有效期限制。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，则代理选择 true，则 CA 会拒绝续订请求。

图 3.5. 在代理服务页面中绕过 CA 限制选项



注意

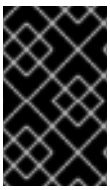
CA Validity Default 仅适用于 CA 签名证书续订。在 CA 有效期间，仍必须发布和更新其他证书。

CA 的单独配置设置 ca.enablePastCATime 可用于允许通过 CA 的有效周期续订证书。但是，这适用于 *该* CA 发布的每个证书。由于潜在的安全问题，不建议在生产环境中使用此设置。

3.7. 管理主题名称和主题备用名称

证书的主题名称是 可分辨名称(DN)，包含识别签发证书的实体的信息。此主题名称可以从标准 LDAP 目录组件构建，如通用名称和组织单元。这些组件在 X.500 中定义。除了 - 甚至替换主题名称外，证书还可以具有 *主题替代名称*，这是为包含 X.500 中未定义的附加信息的证书设置的扩展类型。

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，则必须存在 Subject Alternative Name 扩展并标记为 critical。

3.7.1. 在 Subject Name 中使用 Requester CN 或 UID

证书请求的 *cn* 或 *uid* 值可用于构建发布的证书的主题名称。本节演示了一个配置集，它在证书请求中存在 Subject Name Constraint 中需要指定 naming 属性(CN 或 UID)。如果缺少 naming 属性，则请求将被拒绝。

此配置有两个部分：

- CN 或 UID 格式在 Subject Name Constraint 的模式 配置中设置。
- 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 中设置。

例如，要在主题 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，如果请求与 CN 为 cn=John Smith，则证书将通过主题 DN cn=John Smith,DC=example, DC=com 签发。如果请求位于 中，但 UID 为 uid=jsmith，且没有 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” 中介绍。

3.7.2. 将 LDAP 目录属性值和其他信息插入到主题 Alt 名称

可以使用 Subject Alt Name Extension Default 配置中的匹配变量，将来自 LDAP 目录或请求者提交的信息插入到证书的主题备用名称中。此默认设置信息的类型（格式），然后设置用于检索信息的匹配模式(variable)。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
```

这会插入请求者的电子邮件作为主题 alt 名称中的第一个 CN 组件。要使用其他组件，请按顺序递增 Type_、Pattern_ 和 Enable_ 值，如 Type_1。

配置主题 alt 名称也会在 第 B.1.23 节 “主题备用名称扩展默认值” 中详细介绍。

将 LDAP 组件插入到证书的主题 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。

注意

pkiconsole 已被弃用。

有关设置 **CMC** 共享令牌的详情，请参考 第 10.4.2 节 “设置 CMC 共享 Secret”。

2. **ldapStringAttributes** 参数指示身份验证插件从用户的 **LDAP** 条目读取 ***mail*** 属性的值，并将该值放在证书请求中。当值位于请求中时，证书配置文件策略可以设置为插入扩展值的值。

第 B.2.11 节 “主题名称约束” 和 第 B.1.27 节 “主题名称默认” 中涵盖了 ***dnpattern*** 参数的格式。

3. 要让 **CA** 在证书扩展中插入 **LDAP** 属性值，请编辑配置文件的配置文件，并为扩展插入策略设置参数。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 “以 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将添加 Subject Alternative Name 扩展，并带有请求者 *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
```

有很多属性可以通过在策略集中的任何 Pattern_ 参数中设置为令牌(*X*)自动插入到证书中。常见的令牌在 表 3.1 “用于保留证书的变量” 中列出，默认配置集包含如何使用这些令牌的示例。

表 3.1. 用于保留证书的变量

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(***mail***)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。
$request.requestor_email$	提交请求的人员的电子邮件地址。
$request.requestor_name$	提交请求的人员。
$request.upn$	Microsoft UPN。它具有格式 *(UTF8String) 1.3.6.1.4.1.311.20.2.3,$request.upn$*。

策略设置令牌	描述
$server.source$	指示服务器在主题名称中生成版本 4 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。
$request.subject$	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. 在 SAN 扩展中使用 CN 属性

多个客户端应用程序和库不再支持使用 Subject DN 的通用名称(CN)属性进行验证，这已在 RFC 2818 中被弃用。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 CN 根据 RFC 1034 第 3.5 节，并且具有多个组件时，才会复制 CN。此外，还会保留现有的 SAN 值。例如，基于 CN 的 *dNSName* 值附加到现有的 SAN 中。

要将证书系统配置为使用 SAN 扩展中的 CN 属性，请编辑用于发布证书的证书配置文件。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. **使用配置集的唯一设置号添加以下配置。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. 将新策略集号附加到 *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

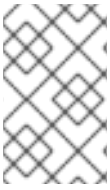
所有默认服务器配置文件都包含 *commonNameToSANDefaultImpl* 默认。

3.7.4. 接受 CSR 的 SAN 扩展

在某些环境中，管理员希望在证书签名请求(CSR)中指定主题备用名称(SAN)扩展。

3.7.4.1. 配置配置文件以从 CSR 中检索 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 Supplied 扩展默认”。



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 工具生成带有两个 SAN 的 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 5.5.2 节 “CMC 注册过程” 中描述的步骤完成 CMC 注册。

第 4 章 设置 KEY ARCHIVAL 和恢复

有关密钥归档和恢复的更多信息，请参阅 *Red Hat Certificate System 规划、安装和部署指南中的归档、恢复和轮转密钥* 部分。

本章解释了如何设置密钥恢复授权(KRA)，之前称为 数据恢复管理器(DRM)，以归档私钥并恢复加密数据的归档密钥。

注意

本章仅讨论通过客户端生成密钥归档密钥。服务器端密钥生成和存档（无论是通过 TPS 启动还是通过 CA 的最终用户实体门户启动）。

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”。

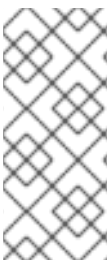
有关 CA 的 EE 门户提供的服务器端密钥生成的详情，请参考 第 5.2.2 节 “使用服务器端密钥生成 CSR”。

注意

Gemalto SafeNet LunaSA 仅支持 CKE 中的 PKI 私钥提取 - 密钥导出模型，且仅在非 FIPS 模式中。FIPS 模式中的 LunaSA Cloning 模型和 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 配对。目前，它被配置为归档和恢复私钥。但是，如果 KRA 证书由外部 CA 发布，而不是安全域中的一个 CA，则必须手动设置密钥归档和恢复过程。

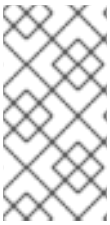
如需更多信息，请参阅 *Red Hat Certificate System 规划、安装和部署指南中的 手动设置密钥存档 部分*。



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 Certificate System Planning, Installation and Deployment Guide 中的 Updating CA-KRA Connector Information after Cloning 部分*。

4.1. 在控制台中配置代理应用密钥恢复



注意

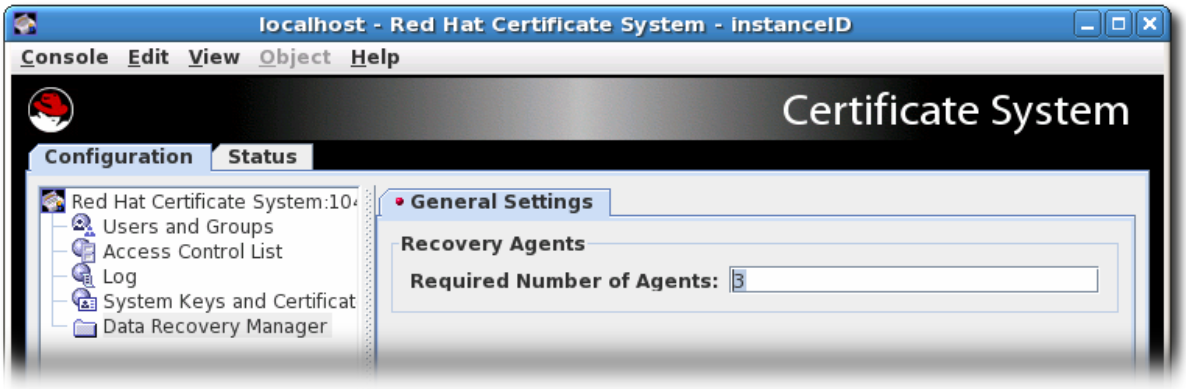
虽然可在控制台中配置密钥恢复代理 *数量*，但要使用的 *组* 只能在 CS.cfg 文件中设置。控制台默认使用 密钥恢复授权代理组。

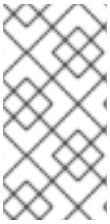
1. 打开 KRA 的控制台。例如：

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

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

3. 在 所需的 Agents 字段中输入用于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密钥恢复的更多信息，请参阅 *Red Hat Certificate System 规划、安装和部署指南中的 命令行配置* 密钥恢复部分。

4.2. 测试密钥 ARCHIVAL 和恢复设置

注意

较新的浏览器不支持浏览器的密钥归档；对于第 1 步，应该为这些浏览器替换 *CRMF 生成客户端*。

测试密钥是否可以成功归档：

1. 使用 CA 的 Manual User Signing and Encryption Certificates Enrollment 表单注册 双证书。

2. 提交请求。登录代理服务页面，并批准请求。

3. 登录到端到端页面，并检查是否已发布了证书。在证书列表中，应该有两个带有连续序列号的新证书。

4. 将证书导入到 Web 浏览器。

5. 确认密钥已存档。在 KRA 的代理服务页面中，选择 Show completed requests。如果密钥成功存档，则会提供有关该密钥的信息。如果没有显示密钥，请检查日志并修正问题。如果密钥已成功存档，请关闭浏览器窗口。

6. 验证密钥。发送签名和加密的电子邮件。收到电子邮件后，打开该邮件并查看其是否签名并加密。消息窗口右上角应当有一个安全图标，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；邮件客户端不应解密邮件。

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，然后单击 Recover Keys 链接。按密钥所有者、序列号或公钥搜索密钥。如果密钥成功存档，则会显示密钥信息。

    b. 点 Recover。

    c. 在出现的表单中，输入与要恢复的私钥对应的 base-64 编码证书；使用 CA 获取此信息。如果通过提供 base-64 编码证书来搜索归档的密钥，则不必在此处提供证书。

d.

确保选择了 Async Recovery 复选框，以便在恢复持续时关闭浏览器会话。

TIP

async 恢复是执行密钥恢复的默认方法。如果要执行同步密钥恢复，则无法关闭浏览器窗口，在恢复过程中无法停止 KRA。

e.

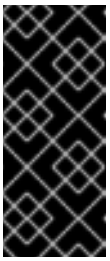
根据代理方案，指定数量的代理必须授权这个密钥恢复。使代理搜索要恢复的密钥，然后批准启动的恢复。

f.

当所有代理都授权恢复后，下一屏幕会请求密码以使用证书加密 PKCS the 文件。

g.

下一屏幕返回一个链接，以下载包含恢复的密钥对的 PKCS the blob。按照链接操作，并将 blob 保存到文件中。



重要

在某些情况下，直接从 gcr-viewer 工具的浏览器中打开 PKCS the 文件可能会失败。要临时解决这个问题，请下载该文件并在 gcr-viewer 中手动打开该文件。

9.

将密钥恢复到浏览器的数据库。将 .p12 文件导入到浏览器和邮件客户端。

10.

打开测试电子邮件。消息应再次显示。

第 5 章 请求、注册和管理证书

最终用户请求并使用证书。虽然证书注册和续订是不仅限于管理员的操作，但了解注册和续订流程可更轻松地管理和创建适当的证书配置文件，如 第 3.2 节 “设置证书配置文件” 所述，并为每个证书类型使用适合的身份验证方法(第 10 章 *注册证书的身份验证*)。

本章讨论请求、接收和更新证书供外部证书系统使用。有关请求和更新证书系统子系统证书的详情，请参考 第 17 章 *管理子系统证书*。

5.1. 关于注册和续订证书

注册 是请求和接收证书的过程。注册过程的机制根据证书类型、生成密钥对的方法以及生成和批准证书本身的方法稍有不同。任何方法、证书注册在高级别上都有相同的基本步骤：

1. 生成证书请求(CSR)。
2. 证书请求提交到 CA。
3. 通过验证请求它的实体并验证请求，并确认请求满足用于提交它的证书配置文件规则。
4. 请求已批准。
5. 请求方检索新证书。

当证书达到其有效期结束时，可以续订。

5.2. 创建证书签名请求

传统上，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，支持通过删除较新版本的浏览器中的密钥生成支持（如 Firefox v69 和 up）来克服出现的不协调。因此，在本节中，我们将不讨论对密钥生成的浏览器支持。虽然无法认为这些浏览器的旧版本不应象旧的 RHCS 文档那样继续正常工作。

从应用程序生成的 CSR 通常采用 PKCS11410 的形式。如果它们被正确生成，则 RHCS 应该支持它们。

在以下小节中，我们将采用 RHCS 支持的以下方法：

- 命令行工具
- 服务器端密钥生成

5.2.1. 使用命令行工具生成 CSR

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

- certutil ：支持创建 PKCS the10 请求。
- PKCS10Client ：支持创建 PKCS the10 请求。
- CRMFPopClient ：支持创建 CRMF 请求。
- pki client-cert-request ：支持 PKCS the10 和 CRMF 请求。

以下小节提供了如何将这些工具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

本节论述了如何使用 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

以下流程描述了如何使用 certutil 工具创建 Elliptic Curve (EC)密钥对和 CSR：

1. 切换到正在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


MIICbTCCAVUCAQAwKDEQMA4GA1UEChMHRXhhbXBsZTEUMBIGA1UEAxMLZXhhb
XBs
  ...
```

这是一个 PKCS the10 PEM 证书请求。

5.2.1.1.2. 使用 certutil 创建带有用户定义的扩展的 CSR

以下流程描述了如何使用 certutil 工具创建带有用户定义的扩展的 CSR。

请注意，注册请求受 CA 定义的注册配置集的限制。请参阅 ???。

1. 切换到正在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. 使用用户定义的密钥用法扩展创建 CSR，以及用户定义的扩展密钥用法扩展，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 Netscape 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 the10 PEM 证书请求。

5.2.1.2. 使用 PKCS10Client创建 CSR

本节论述了如何使用 PKCS10Client 工具创建 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 工具创建 Elliptic Curve (EC)密钥对和 CSR：

1. 切换到正在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 工具为基于 SharedSecret 的 CMC 创建 RSA 密钥对和 CSR。它只将其用于 CMC Shared Secret 验证方法，默认情况下，由 caFullCMCSharedTokenCert 和 caECFullCMCSharedTokenCert 配置集处理。

1.

切换到正在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 中可接受的 CSR 格式，它允许在请求中安全地嵌入密钥归档信息。

本节论述了如何使用 CRMFPopClient 工具创建 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 工具创建 RSA 密钥对和带有 key archive 选项的 CSR：

1. 切换到正在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，请将 -a ec -t false 选项传给命令。

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 工具为基于 SharedSecret 的 CMC 创建 RSA 密钥对和 CSR。它只将其用于 CMC Shared Secret 验证方法，默认情况下，由 caFullCMCSharedTokenCert 和 caECFullCMCSharedTokenCert 配置集处理。

1. 切换到正在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，请将 **-a ec -t false** 选项传给命令。

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** 命令一起使用来生成 CSR。但是，与前面讨论的工具不同，使用 **pki** 生成的 CSR 将直接提交到 CA。可以同时生成 PKCS11410 或 CRMF 请求。

生成 PKCS the10 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
```

成功后将返回请求 ID。

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

例如：

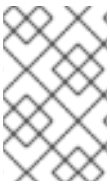
```
pki -d agent token db directory -P https -p 8443 -h host.test.com -c agent token db passwd -n <CA agent cert nickname> ca-cert-request-approve request id
```

如需更多信息，请参阅通过运行 pki client-cert-request --help 命令来 man page。

5.2.2. 使用服务器端密钥生成 CSR

许多新版本的浏览器（包括 Firefox v69 和 Chrome）都删除了生成 PKI 密钥的功能，并支持密钥归档的 CRMF。在 RHEL 上，如 CRMFPopClient （请参阅 CRMFPopClient --help）或 pki （请参阅 pki client-cert-request --help）等 CLI 可以用作临时解决方案。

由于引入令牌密钥管理系统(TMS)，服务器端密钥注册时间已延长，其中密钥可以在 KRA 上生成，而不是本地在智能卡上生成。Red Hat Certificate System 现在采用类似的机制来解决浏览器 keygen 出现问题。密钥在服务器上生成（特别是，在 KRA 上），然后安全地将密钥传输给 PKCS the 中的客户端。



注意

强烈建议您仅将服务器端密钥机制用于加密密钥。

5.2.2.1. 功能高

- 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 提供了启用或禁用密钥归档（例如，*enableArchival* 参数）的选择。

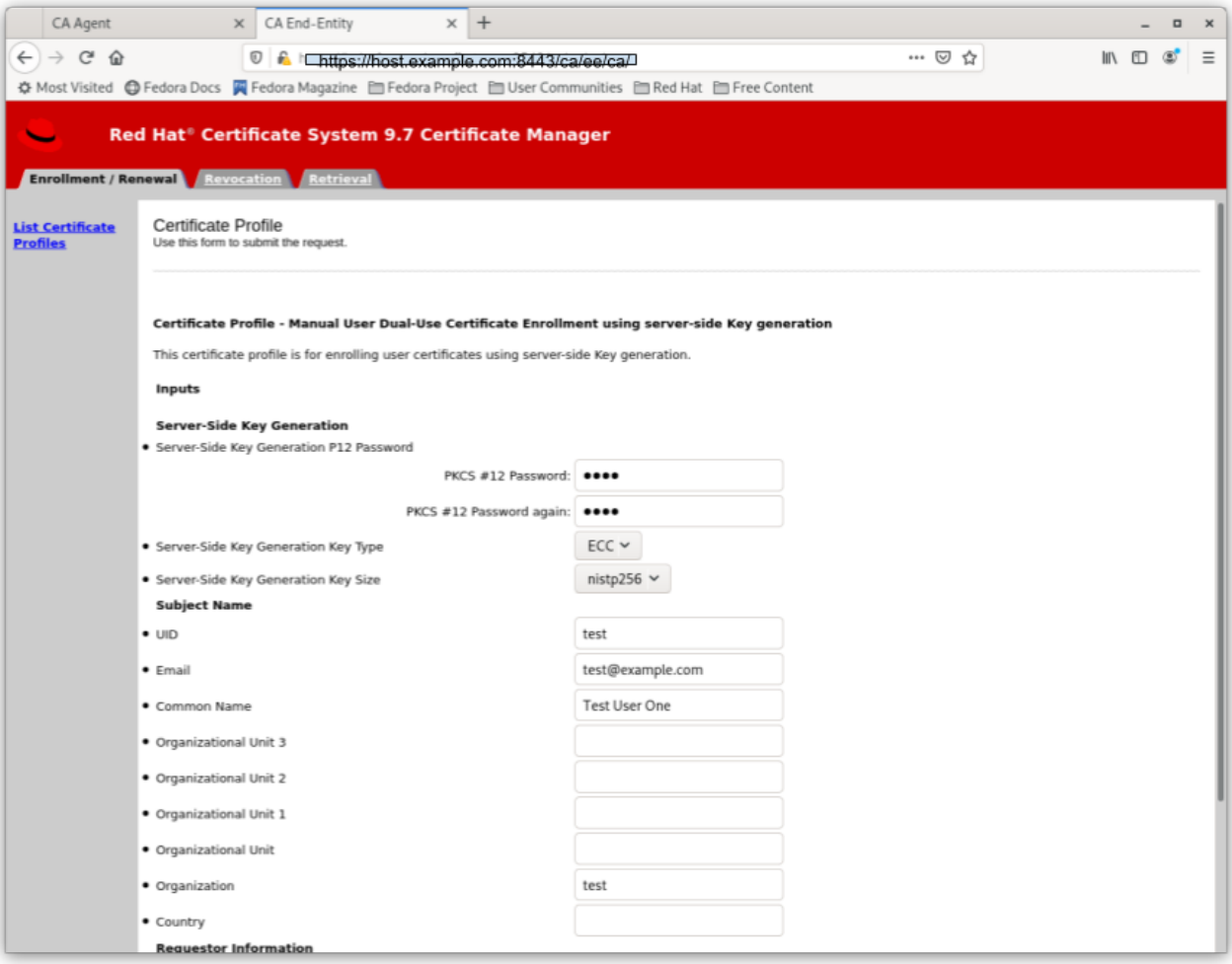
- 支持 RSA 和 EC 密钥
- 支持手动（代理）批准和自动批准（例如，基于密码的目录）

5.2.2.2. 使用服务器端密钥注册证书

默认的 Sever-Side Keygen 注册配置集可在 EE 页面中找到，位于 *List Certificate Profiles* 选项卡下：

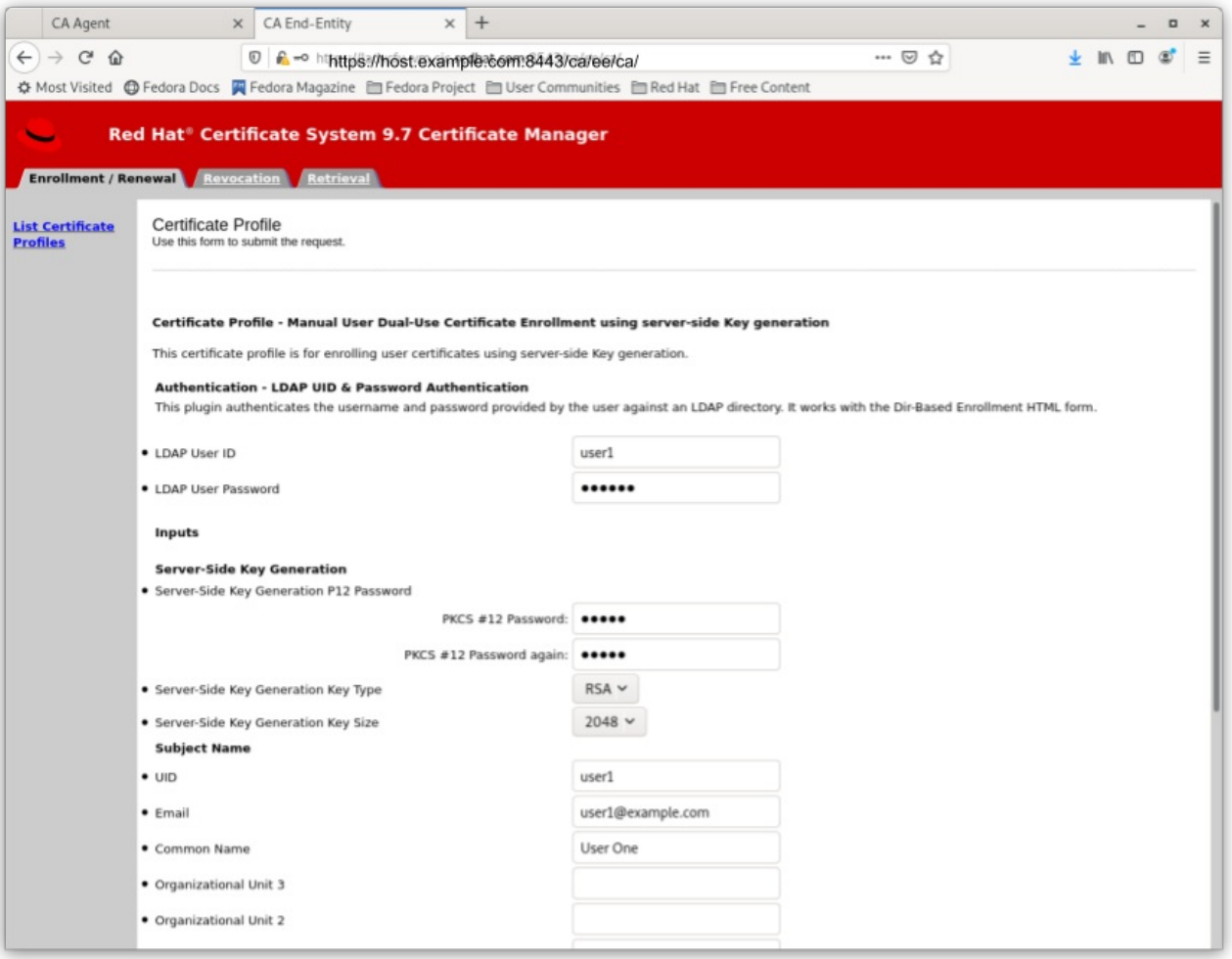
使用服务器端密钥生成手动用户双用途证书注册

图 5.1. 需要代理手动批准的服务器-Side Keygen Enrollment



使用服务器端密钥生成目录验证的用户双用途证书注册

图 5.2. 成功 LDAP uid/pwd 身份验证时自动批准服务器-Side Keygen Enrollment



无论请求是如何批准的，服务器端密钥注册机制都需要最终用户输入 PKCS the package 的密码，该软件包将包含发布的证书以及服务器生成的加密私钥。

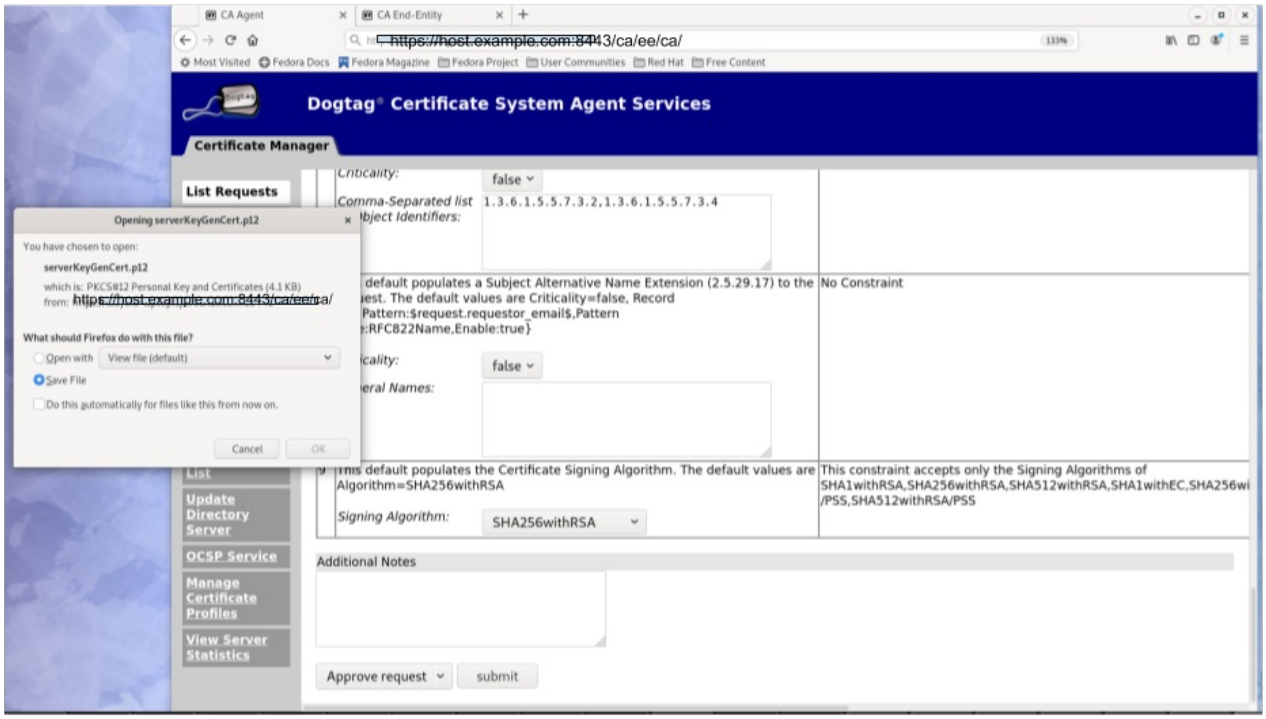
重要

用户不应与任何人共享密码。甚至不是 CA 或 KRA 代理。

当注册请求被批准后，将生成 PKCS the 软件包，

- 如果进行手动批准，PKKKKKM 文件将返回到批准请求的 CA 代理；然后代理应该将 PKCS the 文件转发到用户。
- 如果进行自动批准，PKKKKKKT 将会返回到提交请求的用户

图 5.3. 代理手动批准的注册



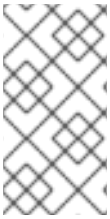
收到 PKCS the 文件后，用户可以使用 CLI （如 pkcs12util ）将此文件导入到每个应用程序自己的用户内部证书/密钥数据库中。例如，用户的 Firefox nss 数据库。

5.2.2.3. 密钥恢复

如果在证书注册配置文件中将 *enableArchival* 参数设置为 *true*，则在 Server-Side Keygen 注册时会存档私钥。然后，授权的 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 请求记录



注意

由于此机制的性质，如果配置集中的 *enableArchival* 参数设置为 *true*，则每个 Server-Side keygen 请求有两个 KRA 请求记录：

- 一个请求类型 *asymkeyGenRequest*

此请求类型不能使用 KRA 代理页面上的 List Requests 来过滤；您可以选择 Show All Requests 来查看列出它们。

- 一个请求类型 *恢复*

5.2.2.4.2. 审计记录

如果启用了，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 （还没有实现）

5.3. 请求和接收证书

如 第 5.1 节 “关于注册和续订证书” 所述，生成 CSR 后，需要将其提交到 CA 以获得颁发。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，有些方法则需要在单独的步骤中提交 CSR，这可由用户执行，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 5.3.1 节 “通过最终用户页面请求和接收证书”
- 第 5.5 节 “使用 CMC 提交证书请求”

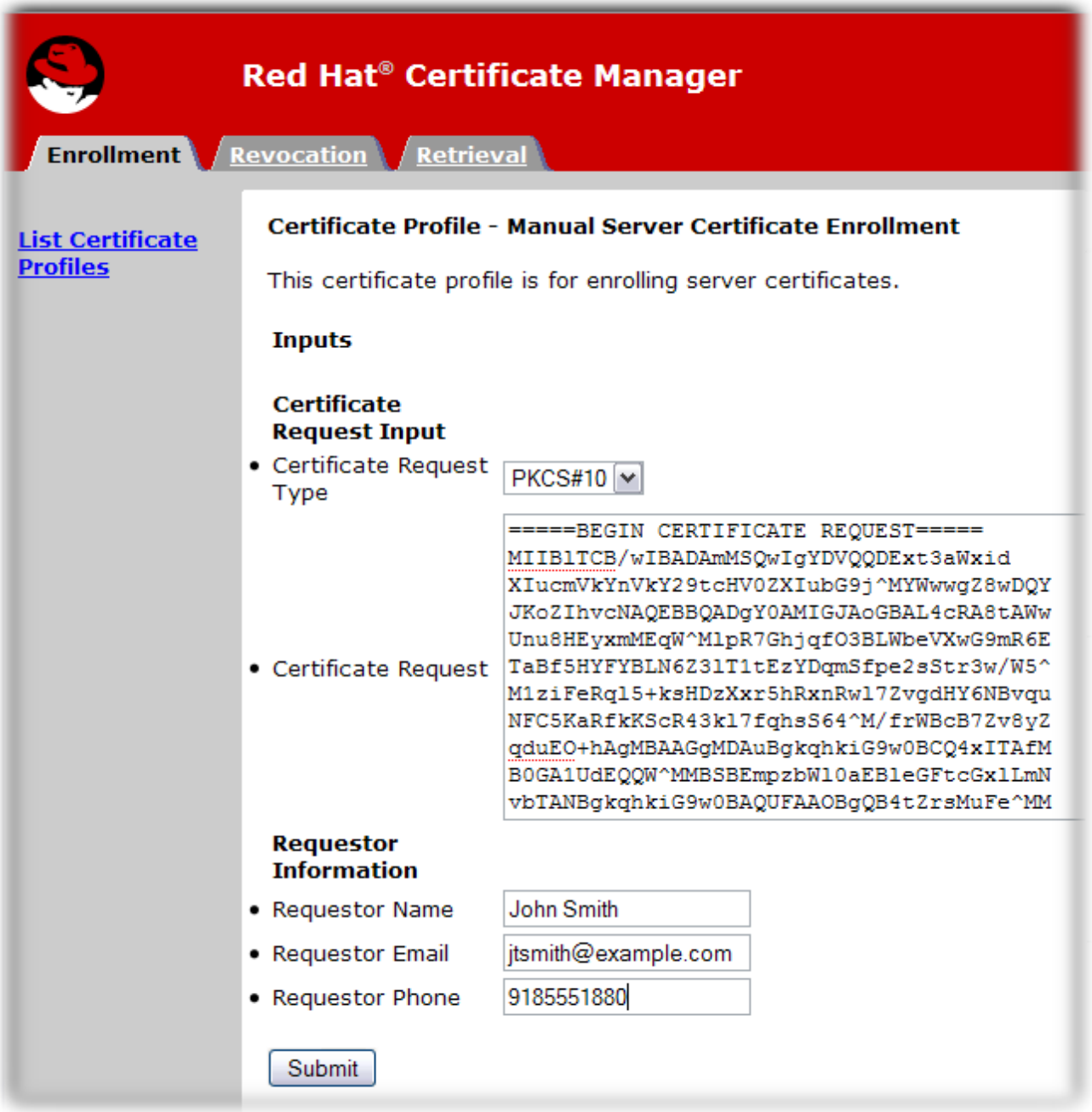
5.3.1. 通过最终用户页面请求和接收证书

在 CA End Entity Portal (i.e. https://*host.domain*:*port*114 /ca/ee/ca)中，最终实体可以使用 Enrollment/Renewal 选项卡下的 HTML 注册表单来提交其证书请求(CSR)，请参阅 第 5.2 节 “创建证书签名请求”。

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

许多默认注册配置集都提供了一个证书请求文本框，其中可粘贴到 Base64 编码 CSR 中，以及 Certificate Request Type 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 the10 或 CRMF。通过子系统管理控制台创建的证书请求是 PKCS04710 ；那些通过 certutil 工具创建的证书请求通常是 PKCS the10。

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

- 请求者名称。这是请求证书的个人的常见名称。

- 请求者电子邮件.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.这是请求者的联系人电话号码。

提交的请求排队以进行代理批准。代理需要处理和批准证书请求。

注意

有些注册配置文件可能允许自动批准，如使用 Red Hat Certificate System 提供的 LDAP uid/pwd 身份验证方法。通过这些配置集注册在下一部分中不需要手动批准。有关支持的批准方法，请参阅 第 10 章 *注册证书的身份验证*。

如果是手动批准，则证书被批准并生成后，您可以检索证书。

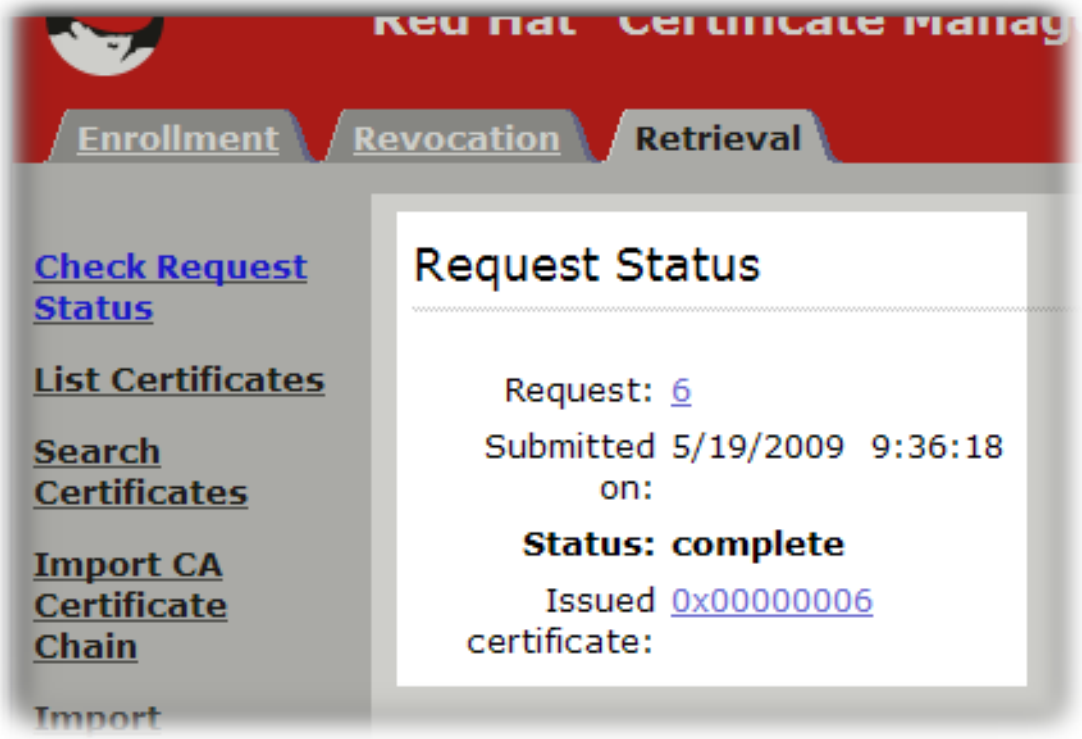
1. 打开 Certificate Manager 端到端页面，例如：

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

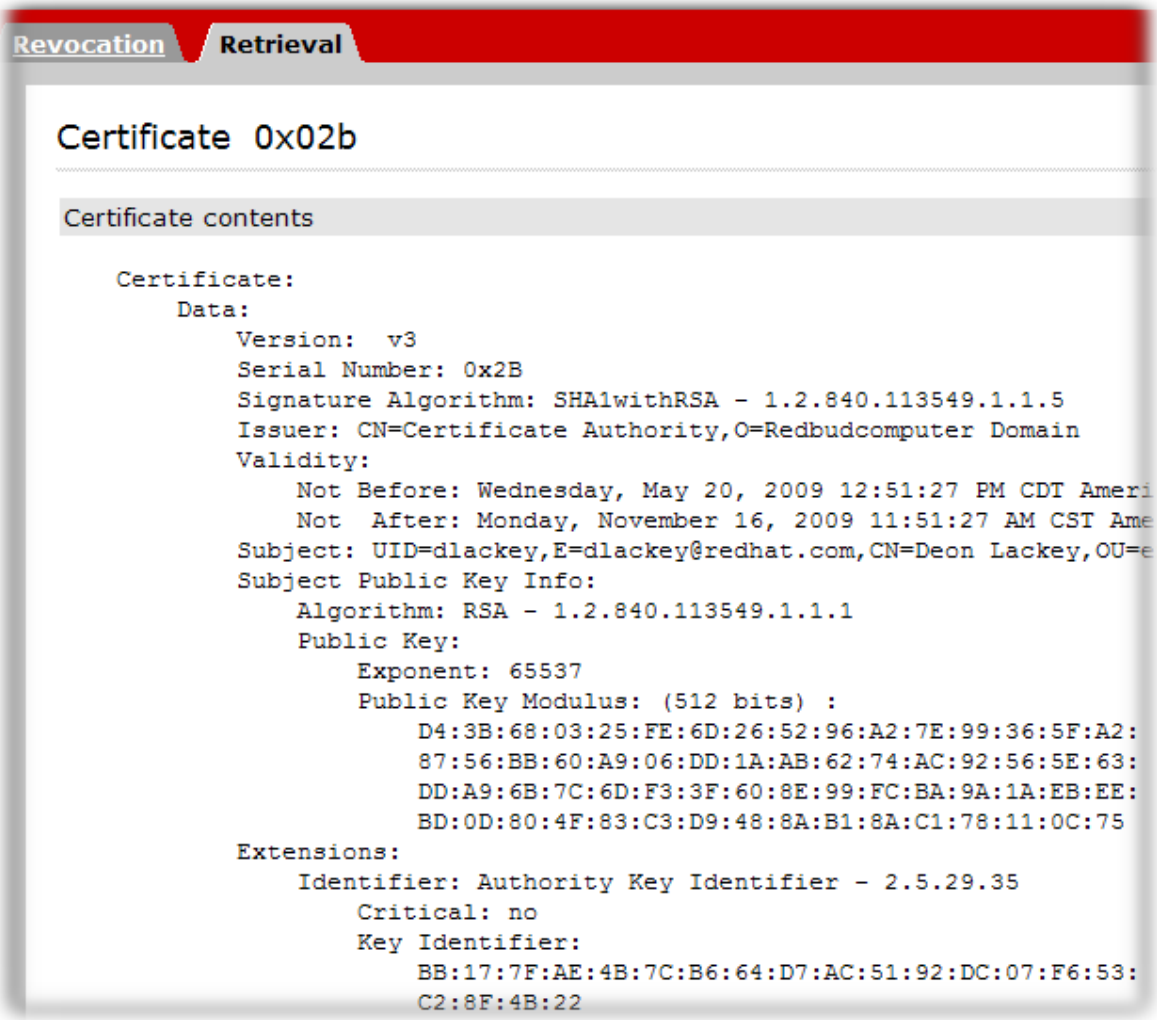
2. 点 Retrieval 选项卡。

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 完成，则有指向证书的链接。点 Issued certificate 链接。



5. 新证书信息以用户为print 格式显示，采用 base-64 编码格式，并且以 PKCS the7 格式显示。



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，请在 Server 部分向下滚动到安装此证书，其中包含 base-64 编码证书。

6. 将 base-64 编码证书（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在的实体的安全模块中。请参阅 第 15.3.2.1 节 “创建用户”。

5.4. 续订证书

本节讨论如何续订证书。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”。

续订证书包括使用与原始证书相同的目的重新生成证书。通常，有两种类型的续订：

- *相同的密钥续订* 取证书的原始密钥、配置集和请求，并使用相同密钥重新创建具有新有效期和过期日期的新证书。这可以通过以下任一方法完成：
  - 通过原始配置文件重新提交原始证书请求(CSR)，或者
  - 使用 certutil 等支持工具使用原始密钥重新生成 CSR
- 重新加密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。*

5.4.1. 相同的密钥续订

5.4.1.1. 重新使用 CSR

最终实体门户上有三种用于同一密钥续订的批准方法。

- *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 成功验证，则会自动批准该证书。*
- *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，并使用相同的证书重新发布。*

5.4.1.1.1. agent-Approved 或基于目录的续订

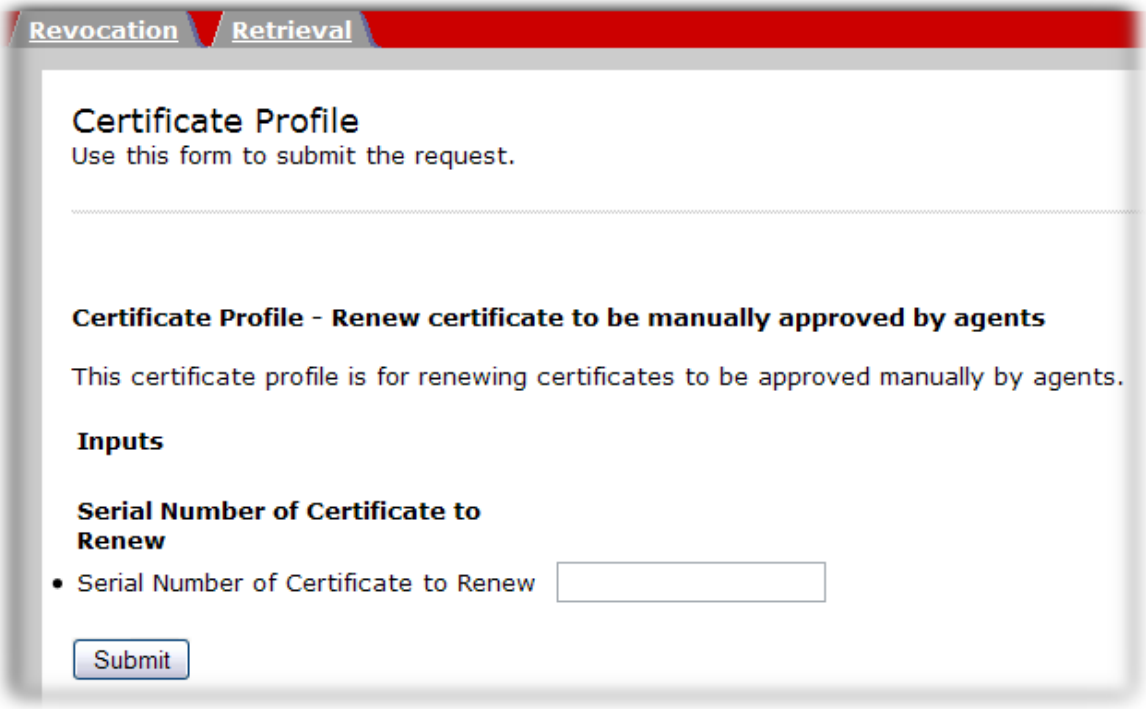
有时，证书续订请求必须手动批准，可以是 CA 代理或为用户目录提供登录信息。

1. *打开签发证书的 CA 的最终服务页面（或克隆）。*

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

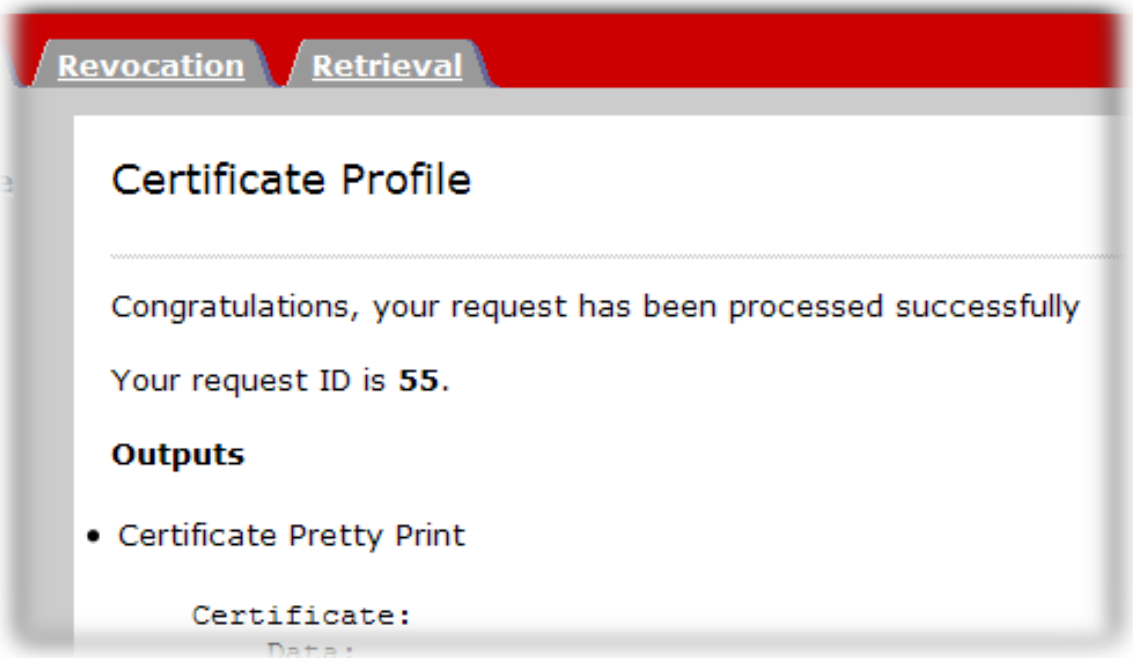
2. *单击要使用的续订表单的名称。*

3. 输入要续订的证书的序列号。这可以采用十进制形式或十六进制形式。



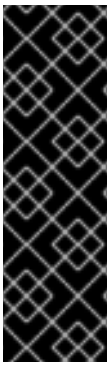
4. 点续订按钮。

5. 请求已提交。对于基于目录的续订，会自动返回更新的证书。否则，续订请求将由代理批准。



5.4.1.1.2. 基于证书的续订

有些用户证书直接存储在您的浏览器中，因此某些续订表单将只检查浏览器证书数据库是否有要续订的证书。如果证书可以被续订，则 CA 会自动批准并重新发布证书。



重要

如果正在续订的证书已经过期，那么它可能无法用于基于证书的续订。浏览器客户端可能会禁止任何使用过期证书的 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。

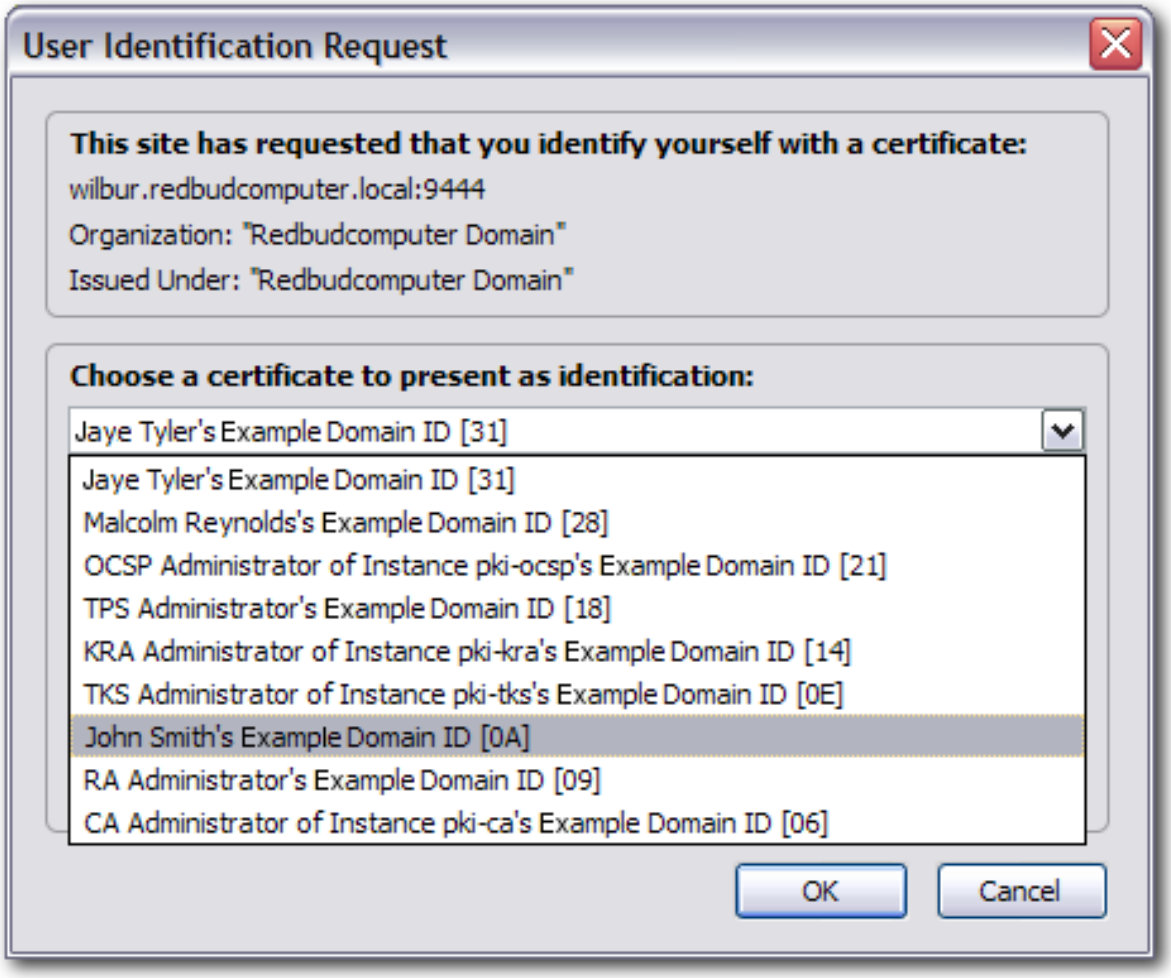
1. 打开签发证书的 CA 的最终服务页面（或克隆）。

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

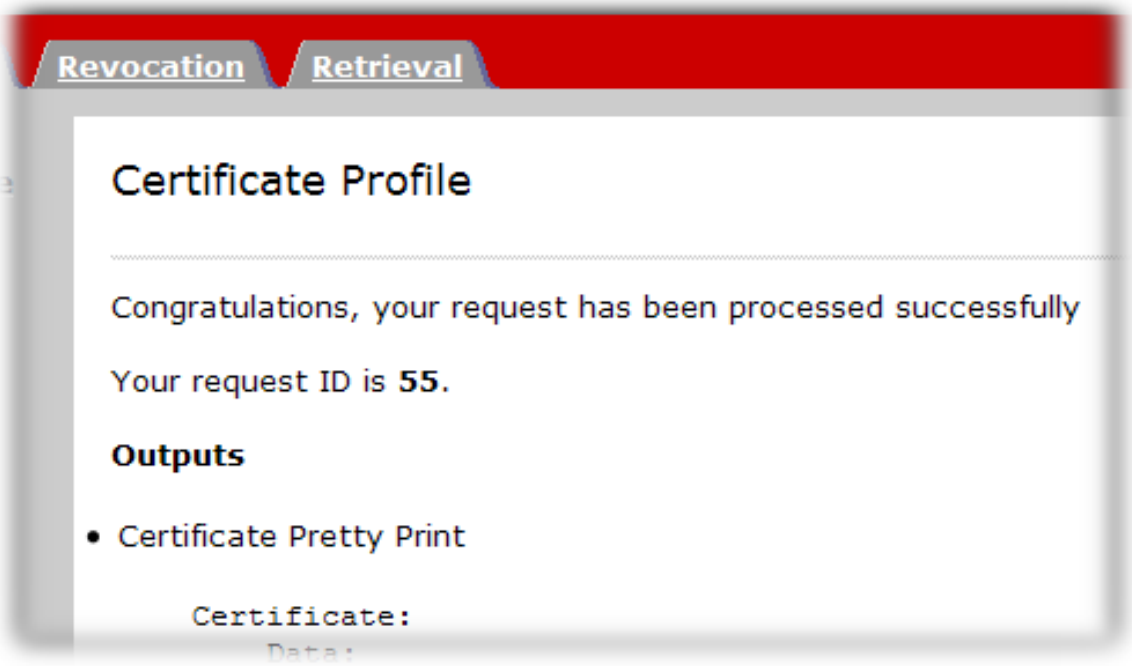
2. 单击要使用的续订表单的名称。

3. 没有输入字段，因此点 续订 按钮。

4. 出现提示时，选择要更新的证书。



5. *请求被提交，并自动返回更新的证书。*



5.4.1.2. 使用同一密钥生成 CSR 续订

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

1. 在 NSS db 中查找对应的密钥 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，如果密钥与 NSS db 中的证书关联，也可以使用 keyid ：

- 使用现有 nickname 生成 CSR：

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

5.4.2. 通过密钥证书续订

因为 重新密钥 续订基本上会生成一个与旧证书相同的信息的新 CSR，只需遵循 第 5.2 节 “创建证书签名请求” 中描述的任何一种方法。请注意，输入与旧证书相同的信息。

5.5. 使用 CMC 提交证书请求

本节描述了通过 CMS (CMC)使用证书管理注册证书的步骤。

有关使用 CMC 配置和注册证书的通用信息，请参考：

- Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Configuring CMC 部分。
- Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Enrolling with CMC 部分。

- *CMCRequest(1) man page*
- *CMCResponse(1) man page*

CMC 注册可以通过各种方式来满足不同场景的要求。第 5.5.2 节 “CMC 注册过程” 红帽证书系统规划、安装和部署指南中的使用 CMC 注册 部分，以及 更多详情。另外，第 5.5.3 节 “实际 CMC 注册场景” 部分可让管理员决定在哪些场景中应使用哪些机制。

5.5.1. 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，该插件使用代理证书预签名证书。当收到使用代理证书签名的有效请求时，证书管理器会自动发布证书。

注意

CMC 注册会被默认启用。除非更改了配置，否则应该不需要启用 CMC 注册身份验证插件或配置文件。

CMCAuth 身份验证插件还为客户端提供 CMC 吊销。CMC 吊销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 吊销可以使用 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 吊销”。

CMC 请求可以通过浏览器端到端表单提交，也可以使用 HttpClient 等工具向适当的配置集发出请求。CMCRequest 工具生成签名证书请求，然后使用 HttpClient 工具或浏览器终端表单提交，以自动和立即注册和接收证书。

CMCRequest 工具具有一个简单的命令语法，所有在 .cfg 输入文件中提供的配置：

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

也可以使用 CMCEnroll 工具创建单个 CMC 注册，其语法如下：

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

这些工具在 CMCEnroll (1) man page 中进行了更详细的描述。

注意

在引号中包含空格的值。

5.5.1.1. 测试 CMCEnroll

1. 使用 certutil 工具创建证书请求。

2. 将 PKCS the10 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，代理证书的证书通用名称为 CertificateManagerAgentsCert，证书数据库的密码为 secret，如下所示：

```
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一个文件中，其文件名相同，且 .out 附加到文件名中。

4. 通过终端实体页面提交签名证书。

   a. 打开 end-entities 页面。

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

   b. 从证书配置文件列表中选择 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. 证书会立即处理并返回。

6. 使用 agent 页面搜索新证书。

5.5.2. CMC 注册过程

使用以下常规流程使用 CMC 请求和发布证书：

1. 使用以下格式之一创建证书签名请求(CSR)：

   - PKCS the10 格式
   - 证书请求消息格式(CRMF)格式

   有关以这些格式创建 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 数据库中。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 Red Hat Certificate System Planning, Installation, installation, and

Deployment Guide 中的 Managing Certificate/Key Crypto Token 部分的指导来验证并导入 admin 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，请确保所有中间证书和 **root CA** 证书都已导入。

- 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** 请求创建一个配置文件，如 **/home/user_name/cmc-request.cfg**，其内容如下：

```
# 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，CMCRequest 工具会将 CMC 请求存储在请求配置文件中的 output 参数中指定的文件中。

5.

*为 **HttpClient** 创建配置文件，如 **/home/user_name/cmc-submit.cfg**，稍后的步骤中使用该文件向 **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

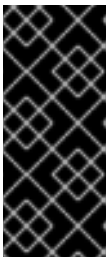
nickname 参数中指定的证书 nickname 必须与之前用于 CMC 请求的证书匹配。

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，此参数中指定的配置集必须使用 CMCAuth 身份验证插件。在用户发起的注册中，配置集必须使用 CMCUserSignedAuth 插件。详情请查看 第 10.3 节 “CMC 身份验证插件”。

7. **向 CA 提交 CMC 请求：**

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

8. **要将 CMC 响应转换为 PKCS the7 证书链，请将 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
```

5.5.3. 实际 CMC 注册场景

本节论述了频繁的实际使用场景及其工作流，以便 CA 管理员能够决定在哪些情况下使用哪些 CMC 方法。

有关使用 CMC 注册证书的通用过程，请参阅 第 5.5.2 节 “CMC 注册过程”。

5.5.3.1. 获取系统和服务器证书

如果服务（如 LDAP 或 Web 服务器）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 的代理以进行批准。这个过程使用 第 5.5.2 节 “CMC 注册过程” 中描述的步骤。另外，请考虑以下要求：

注册配置集

代理必须使用 第 10.3 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 验证机制的自定义配置集。

CMC 签名证书

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

注意

CA 代理必须有权访问自己的私钥。

httpclient TLS Client Nickname

使用同一证书在 CMCRequest 工具的配置文件中签名，与 HttpClient 的配置文件中的 TLS 客户端身份验证相同。

HttpClient servlet Parameter

传递给 HttpClient 实用程序的配置文件中的 servlet 指的是 CMC servlet 和处理请求的注册配置文件。

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

更多详情：

- 当代理预签名 CSR 时，会被视为已建立 Identification，因为代理会检查 CSR 进行验证。不需要额外的 CMC 特定标识验证。
- PKCS the10 文件已经提供 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，因为代理会检查识别。

5.5.3.2. 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 5.5.3.2.1 节 “使用代理证书签名 CMC 请求”。
- 证书注册通过使用 Shared Secret 进行身份验证。请参阅 第 5.5.3.2.2 节 “使用共享 Secret 验证证书注册”。

5.5.3.2.1. 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 5.5.3.1 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。唯一的区别是用户创建 CSR 并将其发送到 CA 代理以进行批准。

5.5.3.2.2. 使用共享 Secret 验证证书注册

当用户想要获取第一个签名证书且代理无法批准请求时，如 第 5.5.3.2.1 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的证书。然后，这个证书可用于为用户的其他证书签名。

在这种情况下，使用 Shared Secret 机制来获取用户的第一个签名证书。将以下信息与 第 5.5.2 节 “CMC 注册过程” 搭配使用：

1. 以用户或 CA 管理员身份创建共享令牌。详情请参阅 Red Hat Certificate System 规划、安装和部署指南中的 共享 Secret 工作流 部分。

请注意：

- 如果用户创建了令牌，用户必须向 CA 管理员发送令牌。
- 如果 CA 管理员创建了令牌，管理员必须与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，将共享令牌添加到 LDAP 中的用户条目中。详情请参阅 Red Hat Certificate System Planning, Installation, installation, installation, and deployment Guide 中的 第 10.4.2.1 节 “将 CMC 共享 Secret 添加到用于证书注册的用户条目中” 和 Enabling the CMC Shared Secret Feature 部分。

3. 在传递给 CMCRequest 工具的配置文件中使用以下参数：

- 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*

5.5.3.3. 获取用户只加密的证书

本节论述了获取使用现有用户签名证书签名的只加密证书的工作流：

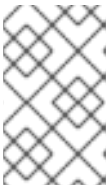
注意

如果用户拥有多个用于不同用途的证书，其中有一个被签名，用户必须首先获取签名证书。用户拥有签名证书后，就可以将其用于参与 Of Origin，而无需设置并依赖 CMC 共享 Secret 机制。

有关获取用户第一个签名证书的详情，请参考 第 5.5.3.2 节 “获取用户的第一个签名证书”。

以用户身份：

1. *使用存储在网络安全服务(NSS)数据库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 the10 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，私钥无法签名。因此，不包含参与 Of Possession (POP)。因

此，注册需要两个步骤：如果初始请求成功，则会导致带有 EncryptedPOP 控制的 CMC 状态。然后，用户使用响应并生成包含 DecryptedPOP 控制的 CMC 请求，并在第二个步骤中提交它。

a.

对于第一步，除了默认参数外，用户还必须在传递给 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*
- *CA 需要 popLinkWitnessV2.enable*
- *CA 需要 popLinkWitnessV2.keyGenAlg*
- *CA 需要 popLinkWitnessV2.macAlg*
- *request.privKeyId*

详情请查看 CMCRequest(1) man page。

响应包含：

- *CMC 加密的 POP 控制*

- 带有 POP 所需的 错误的 CMCStatusInfoV2 控制
- 请求 ID

b. 对于第二个步骤，除了默认参数外，用户还必须在传递给 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。

5.5.3.3.1. 使用 Key Archival 仅包含只加密证书的示例

要使用密钥存档执行注册，请生成一个 CMC 请求，在 CRMF 请求中包含用户的加密私钥。以下流程假设该用户已经拥有签名证书。此签名证书的别名在配置文件中的配置文件中设置。

注意

以下流程描述了与只加密密钥一起使用的双端颁发，这些密钥无法用于签名。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 aresuance 的 CRMFPopClient 工具。

有关在 POP_SUCCESS 中使用 CRMFPoPClient 的说明，请参考 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到的 KRA 传输证书的序列号，将证书存储在文件中。例如，要将带有 12345 序列号的证书存储在 /home/user_name/kra.cert 文件中：

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

3. 使用 CRMFPopClient 工具来：

- 使用密钥归档创建 CSR：

  1. 切换到正在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 传输证书嵌套。例如，要将请求存储在 /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
  ```

  请注意命令显示的私钥的 ID。后续步骤中需要 ID，作为第二个往返的配置文件中的 request.privKeyId 参数的值。

4. 为 CRMRequest 工具创建一个配置文件，如 /home/user_name/cmc.cfg，其内容如下：

```
#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 cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.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，CMCRequest 工具会将 CMC 请求存储在请求配置文件中的 output 参数中指定的文件中。

6. *为 HttpClient 创建配置文件，如 /home/user_name/cmc-submit.cfg，稍后的步骤中使用该文件向 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 cert9.db, key4.db and pkcs11.txt
#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 cert9.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. **向 CA 提交 CMC 请求：**

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

如果命令成功，HTTPClient 实用程序会将 CMC 响应存储在配置文件中 output 参数指定的文件中。

8. **通过将响应文件传递给 CMCResponse 工具来验证响应。例如：**

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

如果第一个往返成功，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. **对于第二个往返，请为 DecryptedPOP 创建配置文件，如 /home/user_name/cmc_DecryptedPOP.cfg，稍后的步骤中使用它。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 cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.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，CMCRequest 工具会将 CMC 请求存储在请求配置文件中的 decryptPopRequestFile 参数指定的文件中。

11.

*为 **HttpClient** 创建配置文件，如 **/home/user_name/decrypted_POP_cmc-submit.cfg**，稍后的步骤中使用它向 **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 cert9.db, key4.db and pkcs11.txt
#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 cert9.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. 向 CA 提交 DecryptedPOP CMC 请求：

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

如果命令成功，HTTPClient 实用程序会将 CMC 响应存储在配置文件中 output 参数指定的文件中。

13. 要将 CMC 响应转换为 PKCS the7 证书链，请将 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
```

或者，要以 PEM 格式显示单个证书，请将 -v 传递给实用程序。

如果第二个往返成功，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
```

5.6. 执行 BULKSUANCE

当管理员需要同时提交和生成大量证书时，可能存在实例。证书系统提供的工具组合可用于向 CA 发送包含证书请求的文件。这个示例流程使用 PKCS10Client 命令生成请求，以及 sslget 命令将请求发送到 CA。

1.

由于此过程被脚本化，因此需要设置多个变量来标识 CA （主机、端口）和用于身份验证的项目（代理证书和数据库）。例如，通过在终端中导出变量来为会话设置这些变量：

```
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

本地系统必须具有有效的安全数据库，其中有一个代理的证书。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，并保存到文件中，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. 停止证书系统实例。

```
pki-server stop instance_name
```

e. 使用 **pk12util** 导入证书。

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

如果过程成功，该命令会打印以下输出：

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

f. 启动证书系统实例。

```
pki-server start instance_name
```

2. 必须设置两个额外变量。用于标识用于处理请求的 **CA** 配置集的变量，以及用于发送 **post** 语句的变量，以提供配置文件表单的信息。

-

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

注意

本例向 caAgentServerCert 配置集提交证书请求（在 post 语句的 profileId 项中标识）。可以使用任何证书配置文件，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（例如，）PCS 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. 使用 sslget 将步骤 4 中创建的批量证书请求文件提交到 CA 配置集接口。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
```

5.7. 在 CISCO 路由器上注册证书

简单的证书注册协议(SCEP)由 Cisco 设计，是路由器用来为路由器注册证书的证书颁发机构（如 CA）的一种方式。

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、请求和检索证书。路由器也可以使用 SCEP 检查待处理请求的状态。

5.7.1. 启用 SCEP 注册

出于安全考虑，在 CA 中默认禁用 SCEP 注册。要允许注册路由器，必须手动为 CA 启用 SCEP 注册。

1. 停止 CA 服务器，以便您可以编辑配置文件。

```
pki-server stop instance_name
```

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。如果没有该参数，则使用 参数添加一行。

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

4. 重启 CA 服务器。

```
pki-server start instance_name
```

5.7.2. 为 SCEP 配置安全设置

通过几个不同的参数，管理员可以设置 SCEP 连接的特定安全要求，如不使用同一证书进行注册身份验证和常规证书注册，或者设置允许的加密算法以防止降级连接强度。这些参数列在 表 5.1 “SCEP 安全性的配置参数” 中。

表 5.1. SCEP 安全性的配置参数

参数	描述
ca.scep.encryptionAlgorithm	设置默认或首选加密算法。
ca.scep.allowedEncryptionAlgorithms	设置以逗号分隔的加密算法列表。
ca.scep.hashAlgorithm	设置默认或首选哈希算法。
ca.scep.allowedHashAlgorithms	设置以逗号分隔的哈希算法列表。

参数	描述
ca.scep.nickname	提供用于 SCEP 通信的证书 nickname。除非设置了此参数，否则默认使用 CA 的密钥对和证书。
ca.scep.nonceSizeLimit	设置 SCEP 请求允许的最大非ce 大小（以字节为单位）。默认值为 16 字节。

为 SCEP 注册设置连接的安全设置：

1. 停止 CA 服务器，以便您可以编辑配置文件。

```
pki-server stop instance_name
```

2. 打开 CA 的 CS.cfg 文件。

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

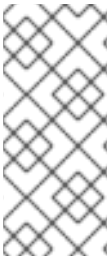
3. 设置所需的安全参数，如 表 5.1 “SCEP 安全性的配置参数” 中列出的。如果该参数不存在，请将其添加到 CS.cfg 文件中。

```
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。

```
pki-server start instance_name
```

5.7.3. 为 SCEP 注册配置路由器



注意

不是所有版本的路由器 IOS 都有相关的加密功能。确保固件镜像具有认证机构互操作性功能。证书系统 SCEP 支持已在运行 IOS C2600 软件(C2600-JK9S-M)、版本 12.2 (40), RELEASE SOFTWARE (fc1)的 Cisco 2611 路由器上测试。

在路由器中注册 SCEP 证书前，请确保正确配置了路由器：

- 路由器必须使用 IP 地址、DNS 服务器和路由信息进行配置。
- 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。

5.7.4. 为路由器生成 SCEP 证书

以下流程详细介绍了如何为路由器生成 SCEP 证书。

1. 选择一个随机 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对 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 10.2.4 节 “配置平面文件身份验证” 中描述了使用平面文件身份验证。

3. 登录路由器的控制台。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 开始，为证书链中的每个 CA 导入 CA 证书。例如，以下命令将链中的两个 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 注册配置文件的 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，请列出路由器中存储的所有证书。**

```
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
```

5.7.5. 使用子 CA

在路由器可以向 CA 进行身份验证前，从 root 开始，必须将 CA 证书链中的每个 CA 证书导入到路由器中。例如，以下命令将链中的两个 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 发行点扩展，请通过将其设置为 可选 来关闭 CRL 要求：

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

之后，设置 CA 身份，如 第 5.7.4 节 “为路由器生成 SCEP 证书” 所述。

5.7.6. 重新注册路由器

在使用新证书重新注册路由器之前，必须删除现有的配置。

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.
```

5.7.7. 启用调试

路由器通过启用 debug 语句在 SCEP 操作过程中提供额外的调试。

```
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
```

5.7.8. 使用 SCEP 发布 ECC 证书

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

- 加密/解密证书 - 指定具有加密/解密功能的 RSA 证书；（以下示例中的scepRSAcert）
- 签名证书 - 获取在客户端中使用的 RSA 证书以签名目的，而不是自签名；（以下示例中的 signingCert 证书）

例如，如果 scepRSAcert 证书是加密/解密证书，并且 signedCert 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'
```

5.8. 使用证书转换

证书系统提供证书转换(CT) V1 支持(rfc 6962)的基本版本。它能够从任何可信日志发布带有嵌入式证书时间戳(SCT)的证书，每个部署站点选择包含其 root CA 证书。您还可以将系统配置为支持多个 CT 日志。这个功能至少需要一个可信的 CT 日志。

重要

部署站点负责建立与可信 CT 日志服务器的信任关系。

有关如何配置证书转换的更多信息，请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Configuring CertificateTransparency 部分。

5.8.1. 测试证书转换

有关如何测试 CT 设置的示例，以下步骤描述了 Google CT 测试日志的实际测试。更为全面的测试过程涉及设置 TLS 服务器，并测试从指定的 CT 日志中包含其证书。但是，以下可作为在签发证书后检查包含 SCT 扩展的快速测试。

测试过程包括生成和提交证书签名请求(CSR)，以便使用 openssl 验证其 SCT 扩展。CS.cfg 文件中的测试配置如下：

```
ca.certTransparency.mode=enabled
ca.certTransparency.log.1.enable=true
ca.certTransparency.log.1.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEw8i8S7qiGEs9NXv
0ZJFh6uuOm<snip>
ca.certTransparency.log.1.url=http://ct.googleapis.com:80/testtube/
ca.certTransparency.log.1.version=1
ca.certTransparency.log.2.enable=true
ca.certTransparency.log.2.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKATl2B3SAbxyzG
OfNRB+AytNTG<snip>
ca.certTransparency.log.2.url=http://ct.googleapis.com:80/logs/crucible/
ca.certTransparency.log.2.version=1
ca.certTransparency.log.3.enable=false
ca.certTransparency.log.3.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEiKfWtuoWCPMEzS
KySjMjXpo38W<snip>
ca.certTransparency.log.3.url=http://ct.googleapis.com:80/logs/solera2020/
ca.certTransparency.log.3.version=1
ca.certTransparency.log.num=3
```

1. **首先，生成一个 CSR，例如：**

```
# PKCS10Client -d . -p passwd -l 2048 -n "cn=user.test.domain.com,OU=user-
TEST,O=TestDomain" -o pkcs10-TLS.req
```

2. **接下来，根据 CS.cfg 中的 ca.certTransparency.mode 参数定义的 CT 模式将 CSR 提交到注册配置集：**

   - **如果参数设为 enabled，则使用任何注册配置集**
   - **如果该参数设为 perProfile，请使用其中一个 CT 配置集：例如 caServerCertWithSCT**

3. **将发布的 b64 证书复制到一个文件中，如 .ct1.pem。**

4. **将 pem 转换为二进制：**

```
#  AtoB ct1.pem ct1.bin
```

5. **显示 DER 证书内容：**

```
#  openssl x509 -noout -text -inform der -in ct1.bin
```

6.

*观察 **SCT** 扩展存在，例如：*

```
    CT Precertificate SCTs:
     Signed Certificate Timestamp:
       Version   : v1 (0x0)
       Log ID    : B0:CC:83:E5:A5:F9:7D:6B:AF:7C:09:CC:28:49:04:87:
           2A:C7:E8:8B:13:2C:63:50:B7:C6:FD:26:E1:6C:6C:77
       Timestamp : Jun 11 23:07:14.146 2020 GMT
       Extensions: none
       Signature : ecdsa-with-SHA256
           30:44:02:20:6E:E7:DC:D6:6B:A6:43:E3:BB:8E:1D:28:
           63:C6:6B:03:43:4E:7A:90:0F:D6:2B:E8:ED:55:1D:5F:
           86:0C:5A:CE:02:20:53:EB:75:FA:75:54:9C:9F:D3:7A:
           D4:E7:C6:6C:9B:33:2A:75:D8:AB:DE:7D:B9:FA:2B:19:
           56:22:BB:EF:19:AD
     Signed Certificate Timestamp:
       Version   : v1 (0x0)
       Log ID    : C3:BF:03:A7:E1:CA:88:41:C6:07:BA:E3:FF:42:70:FC:
           A5:EC:45:B1:86:EB:BE:4E:2C:F3:FC:77:86:30:F5:F6
       Timestamp : Jun 11 23:07:14.516 2020 GMT
       Extensions: none
       Signature : ecdsa-with-SHA256
           30:44:02:20:4A:C9:4D:EF:64:02:A7:69:FF:34:4E:41:
           F4:87:E1:6D:67:B9:07:14:E6:01:47:C2:0A:72:88:7A:
           A9:C3:9C:90:02:20:31:26:15:75:60:1E:E2:C0:A3:C2:
           ED:CF:22:A0:3B:A4:10:86:D1:C1:A3:7F:68:CC:1A:DD:
           6A:5E:10:B2:F1:8F
```

或者，通过运行 asn1 转储来验证 SCT：

```
#  openssl asn1parse -i -inform der -in ct1.bin
```

观察十六进制转储，例如：

```
  740:d=4  hl=4 l= 258 cons:     SEQUENCE
  744:d=5  hl=2 l=  10 prim:      OBJECT            :CT Precertificate SCTs
  756:d=5  hl=3 l= 243 prim:      OCTET STRING      [HEX
DUMP]:0481F000EE007500B0CC83E5A5F97D6B<snip>
```

第 6 章 使用和配置令牌管理系统：TPS 和 TKS

本章提供了使用硬件安全模块（也称为 HSM 或 令牌 ）的步骤来生成和存储证书系统实例证书和密钥。

本章仅包含管理流程。有关令牌管理系统后面的概念的常规信息，请参阅 红帽认证系统规划、安装和部署指南。

6.1. TPS 配置集

注意

有关一般信息，请参阅 *Red Hat Certificate System Planning, Installation and Deployment Guide* 中的 TPS Profiles 部分。

与 CA 注册配置文件（定义并存储在 LDAP 中）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）配置参数采用以下格式设置：

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

在上例中，&lt <explicit op> ;implicit op& gt; 是下面 TPS Operations 部分中讨论的显式和隐式操作之一，& lt;key type > 是为每个证书类型提供的名称。

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 Register (op.enrollö)、format (op.formatö)和 pinReset (op.pinReset)。

隐式操作

隐式操作是 一个操作，它会在处理显式操作时发生令牌的策略或状态。隐式操作包括 keyGen (op.enroll.userKey.keyGenü), renewal (op.enroll.userKey.renewal047), update.applet (op.enroll.userKey.update.applet), 和 key update (op.enroll.userKey.update.symmetricKeys945)。

某些隐式操作根据键类型控制。这包括 恢复、serverKeygen 和 撤销。

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，其密钥的令牌应该在状态转换过程中使用吊销原因 1 吊销认证：

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

根据 RFC 5280，可能的吊销原因及其代码定义如下：

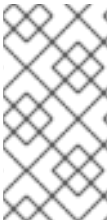
表 6.1. 吊销原因和代码

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

原因	代码
AACompromise	10

6.3. 令牌策略

本节提供了令牌策略列表，它们可使用 TPS UI 按令牌应用。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 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Token Policies 部分。

策略是策略集合，各自用分号(";"")分隔。每个策略都可以使用关键字 YES 或 NO 打开或关闭。以下列表中的每个策略都会被引入其默认值 - 如果策略字符串中没有设置，则 TPS 所执行的操作。

RE_ENROLL=YES

此策略控制令牌是否允许重新注册操作。这允许重新注册令牌（使用证书）并授予新的令牌。如果设置为 NO，服务器将在尝试重新注册时返回错误。

此策略不需要特殊配置。注册将继续执行标准注册配置文件，该配置文件可能最初注册令牌。

RENEW=NO;RENEW_KEEP_OLD_ENC_CERTS=YES

续订允许令牌生成的证书被续订，以放置到令牌中。如果将 RENEW 设置为 YES，则来自企业安全客户端(ESC)的简单注册将导致续订，而不是重新注册。

RENEW_KEEP_OLD_ENC_CERTS 设置决定续订操作是否保留以前的加密证书版本。保留前面的证书后，用户可以访问使用旧证书加密的数据。将这个选项设置为 NO 将意味着任何使用旧证书加密的证书都将无法再恢复。

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使用特定于续订的一些添加的设置，对基本 userKey 标准注册配置集进行了镜像(mirror)。这个奇偶校验是必需的，因为我们开始在续订前，准确续订了最初注册到令牌的证书的数量和类型。

FORCE_FORMAT=NO

如果启用，此策略会导致每个注册操作提示格式操作。这是一个最后一步选项，允许在无需将其返回到管理员的情况下重置令牌。如果设置为 YES，用户发起的每个注册操作将导致格式发生，因此机密将令牌重置为格式化的状态。

不需要额外的配置。发生一个简单的格式时，将执行标准格式操作的同一 TPS 配置文件。

PIN_RESET=NO

此策略决定已经注册的令牌是否可以使用 ESC 执行显式"固定重置"更改。这个值必须设置为 YES，否则尝试的操作将被拒绝，并显示错误。

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 的设置会对所选密码长度进行限制，maxRetries 设置将令牌设置为仅在锁定前允许给定重试次数。

可以使用最新的 TPS 用户界面轻松编辑 TPS 策略。导航到需要更改策略的令牌，再点 Edit。这将启动一个对话框，供您编辑字段，这是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 & lt;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 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Token Policies部分。

æ ¼å¼ï¿½

格式操作(user-initiated)采用一个完全空白状态的令牌，如制造商提供的，并在其上载入 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，并将证书和密钥放在令牌中，以便定制令牌。以下配置示例将解释了如何控制它。

示例显示了不处理续订和内部恢复的基本注册。此处未讨论的设置在 Format 部分中介绍，或者不重要。

```
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” 中讨论了固定重置的配置，因为 pin reset 依赖于一个策略来确定是否被法律执行。

续订

第 6.3 节 “令牌策略” 中讨论了续订的配置，因为续订依赖于一个策略来确定它是否是要执行还是在已经注册的令牌时执行。

恢复

当 TPS 用户界面的用户将一个之前活跃的令牌标记为不可屏蔽的状态，如 "lost" 或 "destroyed" 时，恢复会被隐式设置为 motion。发生这种情况后，同一用户新令牌的下一次注册将遵循以下配置，将用户的旧令牌从用户的旧令牌恢复到此新令牌。

此操作的最终结果是用户会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密钥，以便用户可以根据需要继续加密和解密数据。新的签名证书通常也会放置在这个令牌上，如下例所示。

以下是可手动将令牌放置到 TPS 用户界面中的支持状态列表，如配置中所示：

- tokendb._069=114 - DAMAGED (1): Corresponds 在恢复配置中 销毁。当令牌被物理损坏时使用。
- tokendb._070=114 - PERM_LOST (2): 在恢复配置中将 Corresponds 变为 keyCompromise。当令牌永久丢失时，使用。
- tokendb._071=114 - SUSPENDED (3): Corresponds to onHold in the recovery configuration.当令牌被临时错误替换时，会使用它，但用户需要再次找到它。

- *tokendb._072=114 - TERMINATED (6): Corresponds to terminated in the recovery configuration.用于因为内部原因使令牌不足。*

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执行恢复操作时使用的静态恢复类型（当原始令牌被标记为销毁时）。支持以下方案：

- *恢复 Last (RecoverLast)：恢复要放置到令牌的最新加密证书。*
- *generate New Key and Recover Last (GenerateNewKeyAndRecoverLast): 与 Recover Last 相同，但还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)：生成新的加密密钥并将其放在令牌中。不要恢复任何旧证书。*

例如：

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

以下配置示例决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，以下示例决定了系统对旧令牌上的签名证书执行的操作。在大多数情况下，应使用 GenerateNewKey 恢复方案以避免可能有多个可用的签名私钥副本（例如，在新令牌上恢复，另一个是被永久丢失但被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
```

小程序更新

以下示例演示了如何配置 Coolkey applet 更新操作。此操作可以在格式、注册和 PIN 重置操作期间执行：

```
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 部分中演示。它们提供所需信息来确定是否允许小程序升级、在哪里查找小程序文件，以及用于将令牌升级到的小版本。requiredVersion 中的版本映射到 目录内 的文件名。

密钥更新

此操作可在格式、注册和 PIN 重置操作期间发生，允许用户从制造商提供的默认平台密钥集版本升级其 Global Platform 密钥集版本。

TPS

以下选项将指示 TPS 在代表给定令牌请求下一格式操作期间，将 keyset 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，令牌必须与相同的 TPS 和 TKS 安装一起使用，否则它将会被锁定。

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

您还可以指定一个低于 current 的版本来降级 keyset。

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，新的主密钥标识符 02 必须

映射到 TKS CS.cfg 中的 PKCS the 对象别名，如下例所示：

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

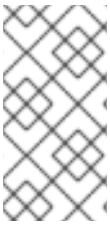
以上会将密钥集号映射到 TKS NSS 数据库中存在的实际主密钥。

主密钥由 ID （如 01 ）标识。TKS 将这些 ID 映射到映射的 masterKeyId 部分中指定的 PKCS the 对象别名。因此，第一个数字被更新，因为主密钥版本已更新，第二个数字保持不变。

当尝试从版本 1 升级到 2 时，映射决定了如何找到主密钥别名，该别名将用于派生新密钥集的 3 部分。

上例中的 internal 设置引用主密钥所在的令牌的名称。它还可能是具有名称（如 nethsm ）的外部 HSM 模块。强大的 new_master 是主密钥 nickname 本身的示例。

6.5. 内部注册



注意

有关一般信息，请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 TPS Profiles 部分。

如果是 内部注册，TPS 配置文件（令牌类型）由 Mapping Resolver 决定。与外部 注册不同，身份验证信息在配置文件本身中定义。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 连接器信息在各个配置集的每个密钥类型下定义。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，您必须先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 配置文件与前面讨论的内部注册配置集类似。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，它允许您选择特定的证书（及其匹配密钥）来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，您必须先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=externalRegAddToToken
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（其默认值）：

```
-
```

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

LDAP 记录属性名称可在此处自定义。确保用户的 LDAP 记录中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，密钥恢复按证书搜索密钥，这使得 < key ID> 值无关。但是，TPS 可以选择性地配置为使用此属性搜索键，因此通常更简单地将值设为 0。该值无效，这可避免检索不匹配的键。

不建议使用密钥 ID 进行恢复，因为在这种情况下，KRA 无法验证证书是否与密钥匹配。

当仅使用证书和 CA 信息指定 certsToAdd 属性时，TPS 假设问题中的证书已存在于令牌中，并且应保留它。此概念称为 Key Retention。

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. 令牌与强制匹配的用户

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中的令牌记录卡-唯一的 ID (CUID)属性匹配。如果记录中缺少此属性(tokencuid)，则不强制执行 CUID 匹配。

```
Tokencuid: a10192030405028001c0
```

关于外部注册的另一个属性是每个令牌上的令牌策略会被绕过。

注意

对于外部注册中要"恢复"的证书和密钥，在用户 LDAP 记录中指定 CA 和 KRA 的连接器信息。与要"恢复"的证书/密钥相关的 TPS 配置文件中指定的任何 CA 和/或 KRA 连接器信息都将被忽略。

```
certstoadd: 59,ca1,0,kra1
```

6.6.5. 委派支持

在验证（登录）、数据加密和解密或签名（例如，公司有一个或多个委托）或签名（限制）方面，委派支持非常有用。

例如，每个委托都有自己的令牌，它们用于代表领导操作。此令牌包含以下证书和密钥的组合（由 TPS 配置文件确定）：

- 身份验证证书/密钥： CN 包含委托的名称和唯一 ID。主题备用名称(SAN)扩展包含领导名称 (UPN)。
- 加密密钥 ：执行加密证书的精确副本。
- 签名证书： CN 包含委托的名称和唯一 ID。SAN 包含参与的 RFC822Name。

使用以下参数启用委派支持：

```
externalReg.delegation.enable=true
```

重要

要临时解决这个问题，请手动将 /var/lib/pki/instance_name/tps/conf/CS.cfg 文件中的 op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId 参数设置为 caTokenUserDelegateAuthKeyEnrollment ：

```
op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

6.6.6. SAN 和 DN 模式

身份验证实例配置中的 auths.instance.<authID > .ldapStringAttributes 指定在身份验证过程中将检索哪些属性。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 记录检索后，可以引用这些属性的值，并用于形成证书的 Subject Alternative Name (SAN)或可辨识名称(DN)，格式为 $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 配置文件中使用模式时，务必要确保正确设置 TPS 配置文件中指定的 CA 注册配置文件。例如：

在 TPS 上，在配置集 delegateIEtoken 中

```
op.enroll.delegateIEtoken.keyGen.authentication.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

在 CA 上，在注册配置集 caTokenUserDelegateAuthKeyEnrollment 中

subjectDNInputImpl 插件必须指定为其中一个输入，以便允许上述 TPS 配置集指定 DN：

```
input.i2.class_id=subjectDNInputImpl
input.i2.name=subjectDNInputImpl
```

同样，要允许由上述 TPS 配置文件指定 SAN，必须指定 subjectAltNameExtInputImpl 插件：

```
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl
```

必须同时指定 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 是用户主体名称(UPN)的 OID， 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 一侧，在 SANpattern 中指定多个 SAN，用逗号分开(''、 '')。在 CA 端，需要以以下格式定义对应的 subjAltExtPattern 数量：

```
policyset.<policy set id>.<policy id>.default.params.subjAltExtPattern_<ordered number>=
```

在上例中， &lt ;ordered number& gt; 以 0 开头，并为每个在 TPS 端指定的 SAN 模式添加一个：

```
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 delegateIEtoken 包含：

- *SAN 特征 ：*

```
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 包含：

```
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. 映射解决程序配置

令牌处理系统默认提供一个映射解析器。解析器称为 FilterMappingResolver。本节将涵盖其配置。

注意

有关 映射解决 的一般信息，请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Mapping Resolver 部分。

6.7.1. Key Set Mapping Resolver

在外部注册过程中，密钥集必须使用解析器解析，然后才能进行身份验证。

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
```

上例定义了名为 0、1 和 2 的三个映射。它们使用示例中的 mappingResolver.keySetMappingResolver.mapping.order=0,1,2 行按顺序排序。这顺序意味着首先针对映射过滤器 0 运行输入参数；只有它们与此过滤器不匹配时，才会尝试映射顺序中的下一个参数。例如，如果评估有以下特征的令牌：

```
CUID=a000000000000000011
appletMajorVersion=0
appletMinorVersion=0
```

然后，它会传递映射 0 并分配其目标（配置为 defKeySet ），因为小程序版本匹配，CUID 不在那个映射的 CUID 启动和结束范围内。

另一方面，如果令牌有以下参数：

```
CUID=b000000000000000000
ATR=2222
appletMajorVersion=1
appletMinorVersion=1
```

在这种情况下，这个令牌失败映射 0， 因为它不在指定的 CUID 范围之外。它还无法映射 1，因为小程序版本匹配，但 ATR 不会。以上令牌将被分配来映射 2 及其目标 jForte。

请注意，映射 2 没有为其任何过滤器分配。这会导致映射与所有令牌匹配，有效使其成为"默认"值。类似的映射必须以映射顺序最后指定，因为永远不会评估后的任何其他映射。

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、pinResetProfileMappingResolver。与上一节中讨论的外部注册问题单相比，内部注册令牌类型实际上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=a000000000
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
```

为上例中的 enrollProfileMappingResolver 定义三个映射。映射名为 0、1 和 2。mappingResolver.enrollProfileMappingResolver.mapping.order=1,0,2 行定义映射将处理的顺序。如果令牌与映射匹配，则不会评估顺序中的进一步映射；如果没有与映射不匹配，则顺序的下一个映射将会被尝试。

如果使用带有以下参数的令牌：

```
CUID=a00000000000000000011
appletMajorVersion=1
appletMinorVersion=0
extension: tokenType=soKey
```

具有此配置的令牌将匹配映射 1 的过滤器，因为 applet 版本匹配，CUID 在指定的 start 和 end 范围内会失败，扩展 tokenType 匹配。因此，此令牌将被分配该映射的目标 - soKey。

在另一个情况下，如果令牌有以下参数：

```
CUID=b00000000000000000010
appletMajorVersion=1
appletMinorVersion=1
```

在这种情况下，令牌将失败映射 1，因为 CUID 不在指定的范围之外。然后，它还会失败映射 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所有与之前过滤器不匹配的令牌。

6.8. 身份验证配置

令牌处理系统默认支持使用用户 ID 和密码(UidPwdDirAuthentication)基于目录的身份验证。使用以下模式在 CS.cfg 文件中定义身份验证实例：

```
auths.instance.<auths ID>.*
```

& lt;auths ID > 是 TPS 配置集要引用的身份验证首选项的验证器名称。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,exe
c-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 身份验证实例配置方式与 CA 的 UidPwdDirAuthentication 身份验证实例相似，因为它们都由同一插件处理。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 项目如何在客户端上标记 UID 和密码。这由上例中的 auths.instance.ldap1.ui.id.UID.name.en=LDAP 用户 ID 和 auths.instance.ldap1.ui.id.PASSWORD.name.en=LDAP 密码 参数控制；此配置告知客户端将 UID/密码对显示为"LDAP 用户 ID"和"LDAP 密码"。这两个参数均可自定义。

credMap.authCred 条目配置内部身份验证插件如何接受提供给它的信息，而 credMap.msgCred 条目配置此信息如何传递给 TPS。这些字段允许您使用自定义插件实现，并且应保留为默认值，除非您使用自定义身份验证插件。

第 6.6 节 “外部注册” 中讨论了与外部注册相关的参数。

与 CA 身份验证配置类似，您可以为同一身份验证实施定义多个身份验证实例。当 TPS 提供多个用户组时，这很有用；您可以指示每个组使用自己的 TPS 配置文件，各自配置为使用自己的目录服务器身份验证。

6.9. 连接器

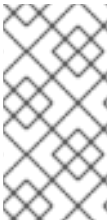
连接器定义 TPS 与其他子系统通信的方式 - 名称 CA、KRA 和 TKS。通常，这些参数会在 TPS 安装过程中设置。以下是连接器配置示例：

```
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 配置文件根据 ID 引用这些连接器。例如：

```
op.enroll.userKey.keyGen.signing.ca.conn=ca1
```

可以定义同一类型的多个连接器（例如，多个 CA 连接器）。当一个 TPS 实例为不同的令牌组提供多个后端证书系统服务器时，这可能很有用。



注意

目前不支持 TPS 中的连接器自动故障转移。只要正在克隆原始系统的克隆，就必须执行手动故障转移流程，以将 TPS 指向备用 CA、KRA 或 TKS。

6.10. 吊销路由配置

要配置撤销路由，您必须首先定义相关 CA 连接器的列表，并使用以下格式将它们添加到连接器列表中：

```
tps.connCAList=ca1,ca2
```

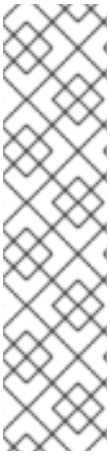
另外，您必须将 CA 签名证书添加到 TPS nssdb 并设置信任：

```
#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 可能会自动计算 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=
```

这是预期的行为。

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恢复授权中心(KRA)生成，它是一个可选的证书系统子系统。需要由 KRA 生成密钥，以便在外部注册时允许恢复丢失或损坏的令牌的密钥。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，要求您指定是否要使用密钥归档。如果确认，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 配置文件特定的参数：

```
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
```

将 serverKeygen.enable=true 选项的 serverKeygen.archive 设置为生效。

重要

LunaSA HSM 不支持 RSA 加密的密钥大小比 2048 位小。

例如，若要配置密钥大小为 2048 位，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：

以下配置用于 TKS 和 KRA （通过 TPS）之间的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 the 令牌、参数 kra.keygen.temporaryPairs、kra.keygen.sensitivePairs 和 kra.keygen.extractablePairs 可以自定义密钥生成选项。这些参数都默认设置为 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测试：

NSS （当使用 FIPS 模式时）：

```
kra.keygen.extractablePairs=true
```

nCipher nShield Connect 6000 （默认情况下不要指定）：

指定 RSA 密钥：

```
kra.keygen.temporaryPairs=true
```

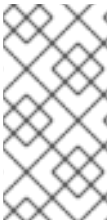
（不要指定任何其他参数。）

生成 ECC 密钥：

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=false
kra.keygen.extractablePairs=true
```

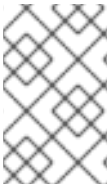
lunasa CKE - 密钥导出模型（非FIPS 模式）：

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=true
kra.keygen.extractablePairs=true
```



注意

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



注意

当 LunaSA CKE - 密钥导出模型处于 FIPS 模式时，无法提取 pki 私钥。

6.12. 设置新密钥集

本节论述了设置令牌处理系统(TPS)和令牌密钥服务(TKS)中设置的默认密钥的替代选择。

TKS 配置

默认密钥集使用 /var/lib/pki/instance_name/tks/conf/CS.cfg 文件中的以下选项在 TKS 中配置：

```
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某些类型或可用于 TMS 中的令牌类的设置。最重要的部分是 3 个开发人员或（开箱即用）会话密钥，用于在对称密钥切换发生前创建安全频道。其他类型的键可能具有这些键的不同默认值。

描述 nistSP800 键离散方法的设置控制此方法是否使用标准 Visa 方法。具体来说，tks.defKeySet.nistSP800-108KdfOnKeyVersion 选项的值决定将使用 NIST 版本。nistSP800-108KdfUseCuidAsKdd 选项允许您在处理过程中使用 CUID 的传统密钥 ID 值。较新的 KDD 值最常被使用，因此默认禁用这个选项（默认为 ）。这可让您配置一个新的密钥集来启用对新密钥类别的支持。

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 静态会话键的不同。

证书系统支持 Giesecke & Devrient (G&D) Smart Cafe 6 智能卡的安全通道协议 03 (SCP03)。要在 TKS 中为这些智能卡启用 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。

确定 TPS 中的 keySet 的主要方法涉及 第 6.7 节 “映射解决程序配置”。如需了解建立这个解析器机制所需的准确设置，请参阅链接部分。

如果没有 KeySet Mapping Resolver，则 TPS 有几个回退方法来确定正确的 keySet ：

- 您可以将 tps.connector.tks1.keySet=defKeySet 添加到 TPS 的 CS.cfg 配置文件中。
- 某些客户端可能会被配置为显式传递所需的 keySet 值。但是，企业级安全客户端目前没有此功能。
- 当 TPS 根据所需方法计算正确的 keySet 时，对 TKS 的所有请求都通过 keySet 值创建安全频道。然后，TKS 可以使用自己的 keySet 配置（上面描述）来确定如何继续。

6.13. 设置新主密钥

本节介绍了在令牌密钥服务(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 Red Hat Certificate System Planning、安装和部署指南。

过程 6.1. 创建新主密钥

1. *获取内部访问 TKS 安全数据库所需的 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 工具生成新的主密钥。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 Wrapped Master 密钥(Key Ceremony)

如果要在外部令牌或多个位置使用主密钥，则必须 嵌套 它，以便安全地将其传送到硬件令牌。tkstool 工具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工具。传输嵌套主密钥的过程通常称为 Key Ceremony。

注意

传输密钥只能与它们生成的主密钥一起使用。

过程 6.2. 生成和传输 Wrapped 主密钥

1. 获取访问 **Token Key Service** 安全数据库所需的内部 **PIN**：

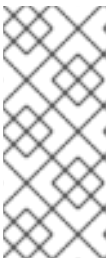
```
# cat /var/lib/pki/pki-tomcat/tks/conf/password.conf

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 工具打印每个生成的三个会话键的密钥共享和 **KCV** 值。将它们保存到文件中，因为有必要在此流程以后在新数据库中重新生成传输密钥，并在丢失时重新生成密钥。

4. 出现提示时，填写数据库密码。然后，按照屏幕说明生成随机 **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。按照屏幕说明进行操作，直到最终信息：

```
Successfully generated, stored, and named the transport key!
```

6. 使用传输密钥生成并嵌套主密钥，并将其存储在名为 **file** 的文件中：

```
# 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. 将嵌套的主密钥复制到适当的位置或工具中。

8. 如有必要，在 **HSM** 或工具中生成新的安全数据库：

```
# 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密钥相同的密钥。以这种方式重新生成传输密钥要求您为此流程前面生成的每个会话密钥输入会话密钥共享和 **KCV**。

```
# tkstool -I -d <directory> -n verify_transport
```

9. 使用传输密钥解压缩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 共享 SYMMETRIC 密钥

共享对称密钥必须存在于 TPS 和 TKS 子系统的 NSS 数据库中。此密钥在创建 TPS 子系统时自动生成。如果 TPS 和 TKS 都在同一个 Tomcat 实例中安装，则不需要额外的设置，因为 TKS 将自动使用由 TPS 创建的密钥；但是，如果这两个子系统都位于独立的实例上，甚至不同的物理主机，您必须遵循本节中的步骤安全地将密钥传送到 TKS。

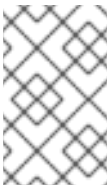
可以使用几种可能的方法在 TPS 和 TKS 之间安全地传输共享密钥：

- authomatic 方法：当将 TPS 的子系统证书保留在软件 NSS 数据库中时，此方法可以正常工作。
- 如果上述方法失败，则可使用回退手动方法，其中使用 tkstool 程序在 TPS 上生成共享密钥，从 TPS 中嵌套密钥，从而在传输中公开密钥，并将其解压缩到 TKS NSS 数据库中。

下面描述了 TPS 和 TKS 的一般配置，无论要用于导入密钥的方法是什么。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，可以扩展上述列表。

TPS

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```

注意

主机名必须与 TKS 端配置的主机名相同。

6.14.1. 手动生成和传输共享分片密钥

这部分论述了如何手动生成和传输共享对称密钥。在自动生成和传输失败时，此方法很有用，但应该避免使用。

manual 方法由两个过程组成。第一个是在 Token Key Service 一端执行，另一个是在令牌处理系统上执行。

过程 6.3. 手动共享 Secret 密钥方法 - TKS 侧

1. 在第一个系统上安装 Token Key Service。有关安装说明，请参阅 Red Hat Certificate System Planning、安装和部署指南。

2.

停止 TKS 服务：

```
#pki-server stop pki-tomcat
```

3.

进入 /var/lib/pki/pki-tomcat/alias 目录，并使用 tkstool 在 TKS 上创建共享 secret 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信息。确保记下此信息，因为稍后需要将其导入 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**：

```
#pki-server start pki-tomcat
```

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 **Red Hat Certificate System 10 规划、安装和部署指南**。

2. 停止 **TPS** 服务：

```
#pki-server stop pki-tomcat
```

3. 进入 **/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，脚本会提示您输入在上述流程生成和嵌套共享密钥时显示的会话密钥共享共享密钥。

4. 在 **TPS** 中配置共享 **secret**：

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```

5. 启动 **TPS** 服务：

```
#pki-server start pki-tomcat
```

6.15. 对不同的 SCP 版本使用不同的 APPLETS

在证书系统中，**/var/lib/instance_name/tps/conf/CS.cfg** 文件中的以下参数指定为每个令牌操作为所有安全频道协议(**SCP**)版本载入哪个小程序：

```
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 版本设置单独的小程序：

```
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
- 注册
- 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，为 SCP03 配置特定的小程序，并为所有其他协议配置不同的小程序：

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 applet。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 applet 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：

   ```
   pki-server restart instance_name
   ```

*有关为 **Giesecke & Devrient (G&D) Smart Cafe 6** 智能卡启用 **SCP03** 的详情，请参考 第 6.12 节 “设置新密钥集”。*

第 7 章 吊销证书并颁发 CRL

证书系统提供了撤销证书以及生成吊销证书列表的方法，称为证书撤销列表(CRL)。本章描述了撤销证书的方法，描述了 CMC 吊销，并提供 CRL 和设置 CRL 的详细信息。

7.1. 关于撤销证书

证书可由最终用户（证书的原始所有者）或证书管理器代理撤销。最终用户可以使用终端实体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的表单来撤销最终用户证书。这两种情况下都需要基于证书的(SSL/TLS 客户端身份验证)。

最终用户只能撤销包含与为身份验证提供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，最终用户可以选择要撤销的证书，也可以撤销列表中的所有证书。最终用户也可以指定附加详情，如每个证书撤销和吊销原因的日期，或整个证书的列表。

代理可以根据一系列序列号或主题名称组件吊销证书。提交撤销请求时，代理会收到它们可从中获取的证书列表。有关代理如何撤销最终用户证书的说明，请参阅 Red Hat Certificate System Planning、安装和部署指南。

批准撤销请求时，证书管理器在其内部数据库中将对应的证书记录标记为撤销，如果配置为这样做，请从发布目录中删除撤销的证书。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需要访问证书的有效性的信息。由于决定证书的有效性的一个因素是其撤销状态，这些应用程序需要知道是否已撤销的证书。CA 负责执行以下操作：

- 如果 CA 收到撤销请求并批准，则撤销证书。
- 使撤销的证书状态可供需要验证其有效状态的第三方或应用程序使用。

每当撤销证书时，证书管理器会自动更新其内部数据库中证书的状态，它会将内部数据库中的证书副本标记为撤销，如果证书管理器被配置为从数据库中删除证书。

传递证书撤销状态的标准方法是发布吊销的证书列表，即一个证书撤销列表(CRL)。CRL 是一个公开可用的证书列表，它已被撤销。

证书管理器可以配置为生成 CRL。通过在 CRL 配置中启用特定于扩展的模块，即可创建这些 CRL 以符合 X.509 标准。服务器通过其 CRL 发布点框架支持标准 CRL 扩展；有关设置 CRL 扩展以进行发布点的更多信息，请参阅 第 7.3.3 节 “设置 CRL 扩展”。证书管理器可在每次撤销证书时生成 CRL，并以定期间隔生成 CRL。如果设置了发布，则 CRL 可以发布到文件、LDAP 目录或 OCSP 响应器。

CRL 由发布 CRL 中列出的证书的 CA 或 CA 已授权的实体发布并数字签名。CA 可以使用单个密钥对为证书签名，以及它发布或两个单独的密钥对，一个用于签名证书，另一个用于签名 CRL。

默认情况下，证书管理器使用单个密钥对来签署它发布的证书及其生成的 CRL。要为证书管理器创建另一个密钥对，并将其专门用于签名 CRL，请参阅 第 7.3.4 节 “将 CA 设置为使用不同的证书来签名 CRL”。

在定义和配置点时，生成 CRL，并在启用 CRL 生成时生成。

启用 CRL 后，服务器会在证书被撤销时收集撤销信息。服务器会尝试将撤销的证书与设置的所有发布点匹配。给定证书可以匹配任何问题点、一个发布点、几个发布点或所有发布点。当已撤销的证书与发布点匹配时，服务器会将有关证书的信息存储在那个发布点的缓存中。

缓存按照复制缓存的时间间隔复制到内部目录中。当达到创建 CRL 的时间间隔时，会从缓存创建一个 CRL。如果为此问题设置了 delta CRL，则在此时也会创建一个 delta CRL。自证书管理器开始收集此信息以来，完整的 CRL 包含所有撤销的证书信息。自上次更新完整 CRL 后，delta CRL 包含所有撤销的证书信息。

完整的 CRL 按顺序编号，如 delta CRLs。完整的 CRL 和 delta CRL 的数字相同；在这种情况下，delta CRL 的数字与 下一个 完整 CRL 相同。例如，如果完整的 CRL 是第一个 CRL，它是 CRL 1。delta CRL 是 Delta CRL 2。CRL 1 和 Delta CRL 2 的数据与下一个完整 CRL 2 合并，后者为 CRL 2。

注意

当对发布点的扩展进行修改时，不会为该问题点使用下一个完整 CRL 创建 delta CRL。创建 delta CRL，其中 第二个 full CRL 会被创建，然后所有后续完整 CRL。

内部数据库仅存储最新的 CRL 和 delta CRL。当每个新的 CRL 都被创建时，旧的 CRL 都会被覆盖。

发布 CRL 后，每次更新到 CRL 和 delta CRL 都会发布到发布设置中指定的位置。发布方法决定了存储多少个 CRL。对于文件发布，使用 CRL 的数字发布到文件的每个 CRL，因此不会覆盖任何文件。对

于 LDAP 发布，发布的每个 CRL 都取代了目录条目中包含 CRL 的属性中的旧 CRL。

默认情况下，CRL 不包含有关撤销过期证书的信息。服务器可以通过为发布点启用该选项来包括撤销的过期证书。如果包含过期的证书，当证书过期时，有关撤销的证书的信息不会从 CRL 中删除。如果没有包含过期的证书，当证书过期时，有关撤销的证书的信息会从 CRL 中删除。

7.1.1. User-Initiated Revocation

当最终用户提交证书撤销请求时，撤销过程的第一步是证书管理器来识别和验证最终用户，以验证用户是否试图撤销自己的证书，而不是属于其他人的证书。

在 SSL/TSL 客户端身份验证中，服务器要求最终用户提供一个与要撤销的主题名称相同的证书，并使用该证书进行身份验证。服务器通过将为客户端身份验证的证书中的主题名称映射到其内部数据库中的证书中的主题名称来验证撤销请求的真实性。只有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匹配的有效或过期证书。然后，用户可以选择要撤销或撤销列表中所有证书的证书。

7.1.2. 吊销证书的原因

证书管理器可以撤销它发布的任何证书。通常接受拒绝 CRL 中通常会包括的证书的原因代码，如下所示：

- 0.未指定；不给出特定原因。
- 1.与证书关联的私钥被破坏。
- 2.与签发证书的 CA 关联的私钥被破坏。
- 3.证书的所有者不再与证书的签发者关联，并且不再具有与证书获取的访问权限或不再需要证书的权限。
- 4.另一个证书替换这个证书。

- 5.发布证书的 CA 已设计为操作。

- 6.证书正在保留待处理的进一步操作。它被视为撤销，但将来可能会退出，以便证书处于活动状态并再次有效。

- 8.证书将从 CRL 中删除，因为它已从 hold 中删除。这只在 delta CRL 中发生。

- 9.证书被撤销，因为证书的所有者已撤回。

证书可由管理员、代理和结束实体撤销。具有代理权限的代理和管理员可以使用代理服务页面中的表单撤销证书。最终用户可以使用最终用户界面的撤销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。还需要最终用户向服务器进行身份验证才能撤销证书。

每当撤销证书时，证书管理器都会更新其内部数据库中证书的状态。服务器使用内部数据库中的条目跟踪所有撤销的证书，并在配置时，它通过将其发布到中央存储库来通知其他用户中的证书不再有效。

7.1.3. CRL 颁发点

由于 CRL 可能会增长非常大，因此有几个方法可以最大程度减少检索和交付大型 CRL 的开销。其中一种方法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。此分区被称为 CRL 发布点，即维护所有撤销的证书子集的位置。分区可以基于吊销的证书是 CA 证书，无论是因为特定原因吊销，还是使用特定配置文件发布。每个问题点都由其名称标识。

默认情况下，证书管理器会生成并发布一个 CRL，即 master CRL。发布点可以为所有证书、只针对 CA 签名证书或包括过期证书的所有证书生成 CRL。

定义了问题点后，可以将它们包含在证书中，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 发布点，而不是主 CRL 或主 CRL。由于发布点上维护的 CRL 比 master CRL 小，因此检查撤销状态会更快。

CRL 发行版点可以通过设置 CRLDistributionPoint 扩展来与证书关联。

7.1.4. delta CRLs

可以为任何定义的发布点发布 delta CRL。delta CRL 包含自上次更新到完整 CRL 后撤销的任何证书的信息。问题点的 delta CRL 通过启用 DeltaCRLIndicator 扩展来创建。

7.1.5. 发布 CRL

证书管理器可以将 CRL 发布到文件、兼容 LDAP 的目录或 OCSP 响应者。证书管理器中会发布 CRL 的频率，如 第 9 章 发布证书和 CRL 所述。

由于 CRL 可能非常大，发布 CRL 可能需要很长时间，且进程可能会中断。可将特殊发布者配置为通过 HTTP1.1 将 CRL 发布到文件，如果进程中断，CA 子系统的 Web 服务器可能会在其中断时恢复发布，而不必再次开始。这在 第 9.8 节 “设置可恢复的 CRL 下载” 中进行了描述。

7.1.6. 证书撤销页面

证书管理器的末尾级页面包含默认 HTML 表单，用于撤销由 SSL/TLS 客户端进行身份验证的撤销。表单可从 Revocation 选项卡中访问。您可以通过点 User Certificate 链接来查看此类撤销的表单。

要更改表单外观以适应机构的需求，请编辑 UserRevocation.html，该表单允许 SSL/TSL 客户端对客户端或个人证书的撤销撤销。该文件位于 /var/lib/instance_name/webapps/subsystem_type/ee/subsystem_type 目录中。

7.2. 执行 CMC 吊销

与 CMS (CMC)注册上的证书管理类似，CMC 吊销允许用户设置撤销客户端，并使用代理证书或使用匹配的 subjectDN 属性签署撤销请求。然后，用户可以向证书管理器发送已签名请求。

或者，也可以使用 Shared Secret Token 机制进行身份验证 CMC 撤销。详情请参阅 启用 CMC 共享 Secret 功能。

无论用户或代理是否签署请求还是是否使用 Shared Secret Token，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吊销证书”。

- *CMCRevoke.详情请查看 第 7.2.2 节 “使用 CMCRevoke吊销证书”。*

重要

红帽建议使用 CMCRequest 工具来生成 CMC 撤销请求，因为它提供了比 CMCRevoke 更多选项。

7.2.1. 使用 CMCRequest吊销证书

使用 CMCRequest 吊销证书：

1. *为 CMC 撤销请求创建一个配置文件，如 /home/user_name/cmc-request.cfg，其内容如下：*

```
#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 cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.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，CMCRequest 工具会将 CMC 请求存储在请求配置文件中的 output 参数中指定的文件中。

3. *创建配置文件，如 /home/user_name/cmc-submit.cfg，稍后的步骤中使用该文件向 CA 提交 CMC 撤销请求。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.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 cert9.db, key4.db and pkcs11.txt
#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 cert9.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 撤销请求被签名，请将 secure 和 clientmode 参数设置为 true，并填写 nickname 参数。

4. 根据谁签署了请求，必须相应地设置 HttpClient 配置文件中的 servlet 参数：

- 如果代理签署了请求，请设置：

```
servlet=/ca/ee/ca/profileSubmitCMCFull
```

- 如果用户签署了请求，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 吊销证书的详情，请查看 CMCRequest(1) man page。

7.2.2. 使用 CMCRevoke吊销证书

CMC 吊销工具 CMCRevoke 用于通过代理的证书为撤销请求签名。此工具只需传递所需的信息 - 证书序列号、签发者名称和吊销原因 - 用于识别要撤销的证书，然后要求信息来标识执行撤销的 CA 代理（证书别名和数据库）。

证书被撤销的原因可以是以下任意一种（数字是传递给 CMCRevoke 工具的值）：

- 0 - 未指定
- 1 - 密钥已被破坏

- 2 - CA 密钥已被破坏
- 3 - 员工的关联变化
- 4 - 证书已被取代
- 5 - 停止操作
- 6 - 证书处于冻结状态

命令行工具指南中详细介绍了可用的工具 参数。

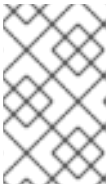
7.2.2.1. 测试 CMCRevoke

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为 ~jsmith/.mozilla/firefox/，则证书的 nickname 为 AgentCert，证书的序列号为 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。

重要

在参数及其值之间没有空格。例如，提供序列号 26 是 -s26，而不是 -s 26。

2. 打开 end-entities 页面。

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

3. 选择 Revocation 选项卡。

4. 选择菜单中的 CMC Revoke 链接。

5. 将 CMCRevoke 的输出粘贴到文本区域中。

6. 从粘贴内容中删除 -----BEGIN NEW CERTIFICATE REQUEST----- 和 ----END NEW CERTIFICATE REQUEST-----。

7. 点 Submit。

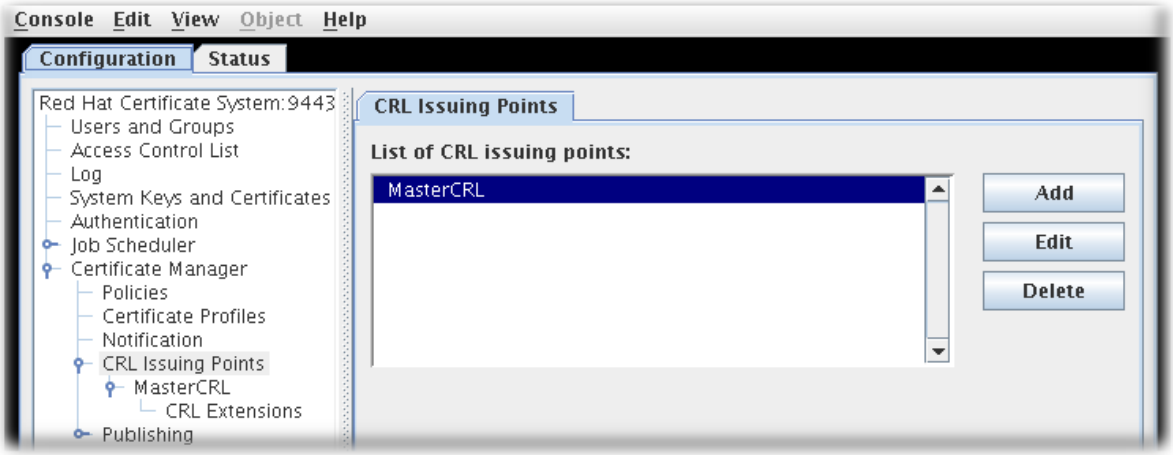
8. 返回后的页面应确认已撤销正确的证书。

7.3. 发布 CRL

1. 证书管理器使用其 CA 签名证书密钥来签署 CRL。要将单独的签名密钥对用于 CRL，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。请参阅 第 7.3.4 节 “将 CA 设置为使用不同的证书来签名 CRL” 了解更多信息。

2. 设置 CRL 发布点。为 master CRL 设置并启用问题点。

图 7.1. 默认 CRL 颁发点



可以创建 CRL 的额外发布点。详情请查看 第 7.3.1 节 “配置颁发点”。

发布点可以创建五类 CRL，具体取决于配置发布点时设置的选项来定义 CRL 将列出的内容：

- Master CRL 包含从整个 CA 中撤销的证书列表。
- ARL 是一个仅包含撤销的 CA 证书的授权撤销列表。
- 带有过期证书的 CRL 包括 CRL 中已过期的证书。
- 来自证书配置文件的 CRL 决定基于最初创建证书的配置集来包括撤销的证书。
- 按原因代码的 CRLs 根据吊销原因代码决定撤销的证书。

3. 为每个发布点配置 CRL。详情请查看 第 7.3.2 节 “为每个颁发点配置 CRL”。

4. 设置为发布点配置的 CRL 扩展。详情请查看 第 7.3.3 节 “设置 CRL 扩展”。

5. 通过为该发布点启用扩展，为发布点、Broata CRLIndicator 或 CRL Number 启用扩展来为发布点设置 delta CRL。

6. 设置 CRLDistributionPoint 扩展，使其包含有关发布点的信息。

7. 将 CRL 设置为文件、LDAP 目录或 OCSP 响应器。有关设置发布的详情，请查看 第 9 章 发布证书和 CRL。

7.3.1. 配置颁发点

发布点定义在新 CRL 中包含哪些证书。默认情况下，为 master CRL 创建一个 master CRL 发布点，其中包含证书管理器的所有撤销证书列表。

要创建新发布点，请执行以下操作：

1. 打开证书系统控制台。

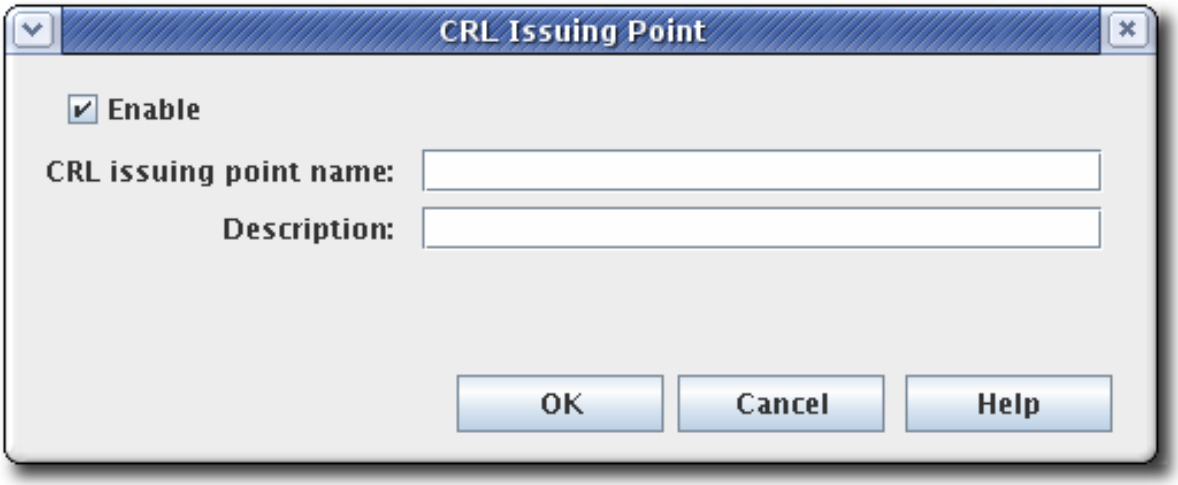
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧导航菜单中展开 Certificate Manager。然后选择 CRL Issuing Points。

3. 若要编辑问题点，请选择发布点，然后单击 Edit。唯一可编辑的参数是发布点的名称，以及发布点是启用或禁用的。

   要添加发布点，请单击 Add。CRL Issuing Point Editor 窗口将打开。

图 7.2. CRL 颁发点编辑器



注意

如果某些字段不足以读取内容，请通过拖动其中一个角来扩展窗口。

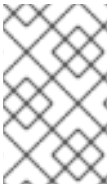
填写以下字段：

- 启用。如果选中，启用问题点；取消选择以禁用。
- CRL 颁发点名称。为发布点指定名称；不允许空格。
- 描述.描述问题点。

4. 点确定。

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。新的问题点列在导航树中 CRL Issuing Points 条目下方。

为新的发布点配置 CRL，并设置与 CRL 搭配使用的任何 CRL 扩展。有关配置发布点的详情，请查看 第 7.3.2 节 “为每个颁发点配置 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 都会出现在代理服务页面的 Update Revocation List 页面中。



注意

pkiconsole 已被弃用。

7.3.2. 为每个颁发点配置 CRL

为发布点配置信息，如生成间隔、CRL 版本、CRL 扩展和签名算法。必须为每个发布点配置 CRL。

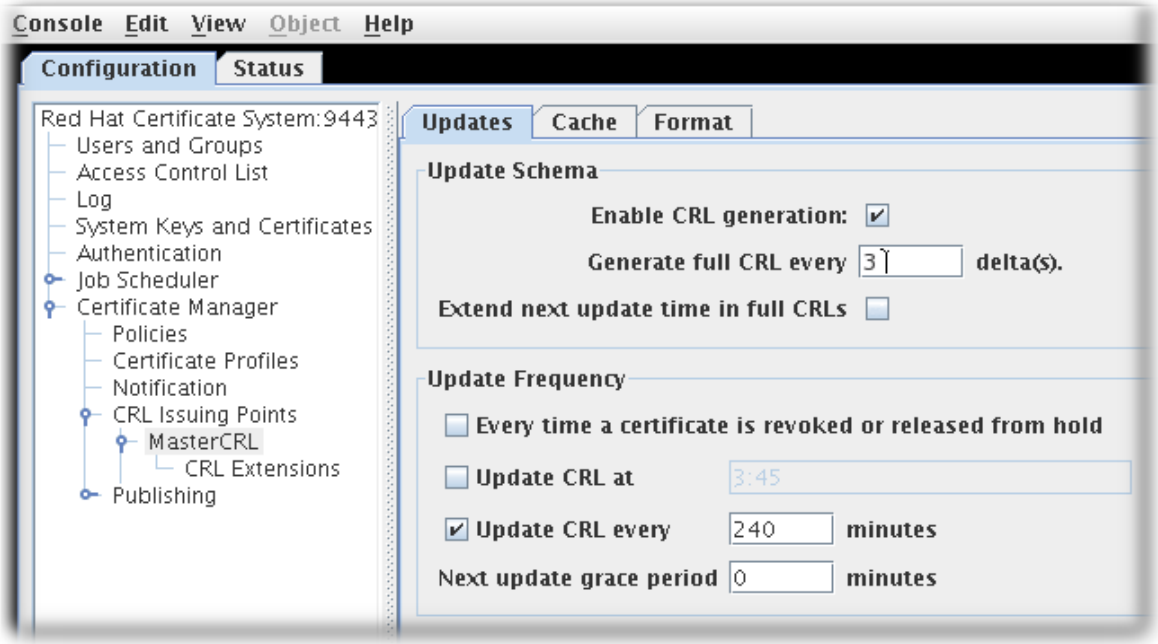
1. 打开 CA 控制台。

```
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** 条目下的发布点名称。

4. 通过在 **Update** 选项卡中提供发布点的信息来配置 CRL 如何和频率更新。此选项卡有两个部分： **Update Schema** 和 **Update Frequency**。



- **Update Schema** 部分有以下选项：

- 启用 CRL 生成。此复选框设定是否为该发布点生成 CRL。

- 生成一个完整的 CRL 增量。此字段设置与更改数量相关的创建 CRL 的频率。

- 在 完整 CRL 中延长下一次更新时间。这提供了一个选项，用于在生成的 CRL 中设置 nextUpdate 字段。nextUpdate 参数显示发布下一个 CRL 时的日期，无论它是完整还是 delta CRL。当使用 full 和 delta CRL 的组合时，在完整 CRL 中启用 扩展下一次更新时间将在完整 CRL 中进行 下一个Update 参数，当下一次完整 CRL 将被发布时，将生成完整的 CRL 参数。否则，完整的 CRL 中的 nextUpdate 参数将在发布下一个 delta CRL 时显示，因为 delta 将是要发布的下一个 CRL。

- Update Frequency 部分在生成 CRL 并发布到目录时设置不同的间隔。

  - 每次从暂停中撤销或发布证书时。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时尝试向配置的目录发出 CRL。如果 CRL 较大，生成 CRL 可能会消耗时间。将证书管理器配置为在每次撤销证书时生成 CRL，可能会持续与服务器联系；在此期间，服务器将无法使用它收到的任何更改来更新目录。

    不建议在标准安装中使用这个设置。应选择此选项以立即测试撤销，例如测试服务器是否向平面文件发出 CRL。

  - 更新位于 的 CRL。此字段会在应更新 CRL 时设置每日时间。要指定多次，请输入逗号分隔列表，如 01:50,04:55,06:55。要输入多个天数的调度，请输入以逗号分隔的列表来在同一天内设置时间，然后是一个分号分隔的列表，以标识不同天数的时间。例如，这会对周期的第 1 天（第 1 天）、50am、4:55am 和 6:55am，第 2 天（第 2 天、5am 和 5pm）进行撤销：

    ```
    01:50,04:55,06:55;02:00,05:00,17:00
    ```

  - 各自更新 CRL。此复选框允许在字段中设置的间隔生成 CRL。例如，要每天发布 CRL，请选中复选框，然后在此字段中输入 1440。

  - 下一次更新宽限期。如果证书管理器以特定频率更新 CRL，则可以将服务器配置为在下次更新的时间具有宽限期，以允许时间创建 CRL 并发出它。例如，如果服务器配

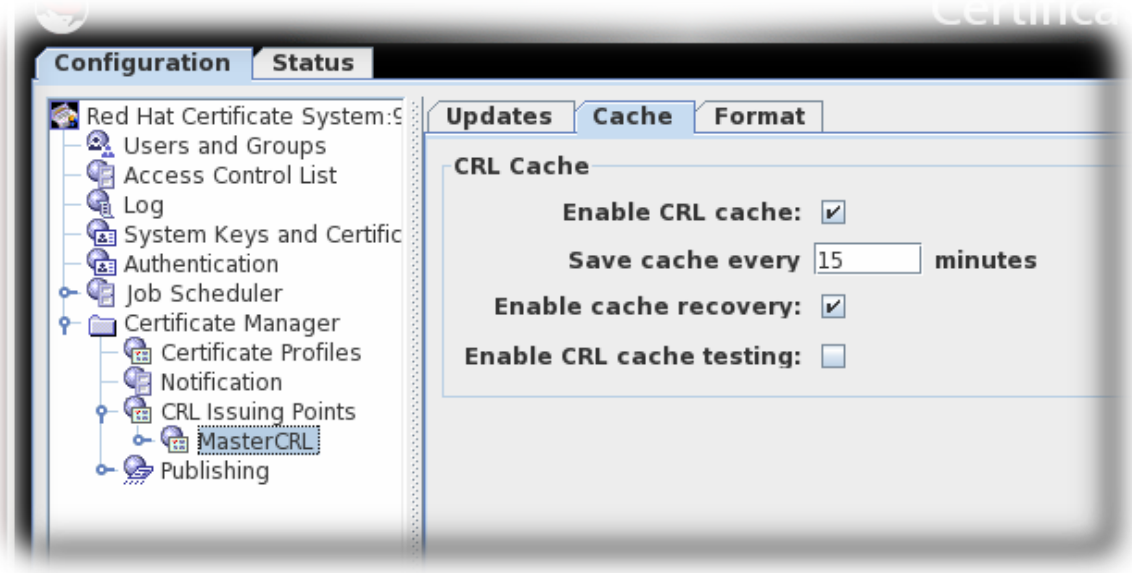
置为每 20 分钟更新 CRL，宽限期为 2 分钟，如果 CRL 在 16:00 更新，则 CRL 会再次更新为 16:18。

重要

由于一个已知问题，当当前设置 full 和 delta Certificate Revocation List 调度时，每次从 hold 选项撤销或发布证书时，更新 CRL 都需要您填写两个 宽限期 设置。因此，要选择这个选项，您需要首先选择 Update CRL per 选项，并为 Next update grace period 一 分钟 输入数字。

5. Cache 选项卡设置缓存是否已启用以及缓存频率。

图 7.3. CRL Cache Tab



- 启用 CRL 缓存。此复选框启用缓存，用于创建 delta CRL。如果禁用了缓存，则不会创建 delta CRLs。有关缓存的详情请参考 第 7.1 节 “关于撤销证书”。

- 按更新缓存。此字段设置缓存写入内部数据库的频率。设置为 0， 以便在每次撤销证书时都会将缓存写入数据库。

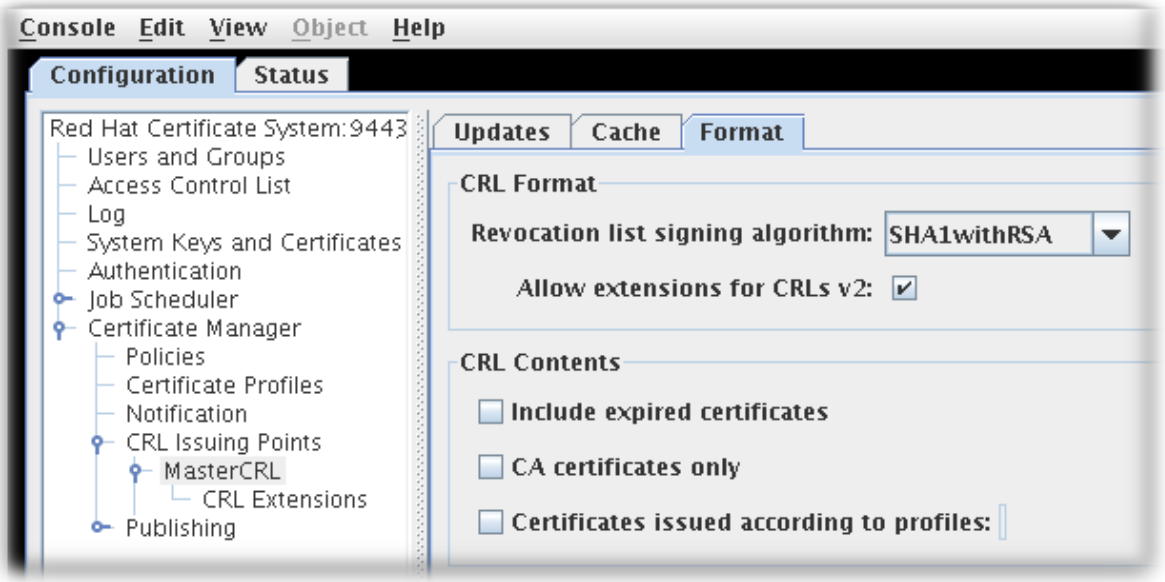
- 启用缓存恢复。此复选框允许恢复缓存。

- 启用 CRL 缓存测试。此复选框为特定 CRL 发布点启用 CRL 性能测试。使用此选项生成的 CRL 不应在部署的 CA 中使用，因为为测试而发布的 CRL 包含只针对性能测试而生成

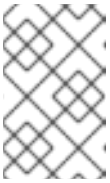
的数据。

6. Format 选项卡设置创建的 CRL 的格式和内容。CRL 格式和 CRL 内容有两个部分。

图 7.4. CRL Format Tab



- CRL Format 部分有两个选项：
  - 吊销列表签名算法 是允许密码加密 CRL 的下拉列表。
  - 允许 CRL v2 的扩展 是一个复选框，它为发布点启用 CRL v2 扩展。如果启用此功能，请设置 第 7.3.3 节 “设置 CRL 扩展” 中描述的所需的 CRL 扩展。



注意

必须打开扩展来创建 delta CRL。

- CRL 内容 部分有三个复选框，用于设置 CRL 中包含的证书类型：

- 包括过期的证书。这包括已撤销的证书。如果启用，证书过期后，有关撤销的证书的信息保留在 CRL 中。如果没有启用此功能，则在证书过期时会删除撤销的证书的信息。

- 仅限 CA 证书。这仅包含 CRL 中的 CA 证书。选择这个选项会创建一个授权撤销列表(ARL)，它只列出撤销的 CA 证书。

- 根据配置文件发布的证书。这只包含根据列出的配置集发布的证书；要指定多个配置集，请输入以逗号分隔的列表。

7. 点击 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”。

注意

pkiconsole 已被弃用。

7.3.3. 设置 CRL 扩展

注意

只有为有问题的点选择了 Allow extensions for CRLs v2 复选框，则仅需要为发布点配置扩展。

创建问题点时，会自动启用三个扩展：CRLReason、InvalidityDate 和 CRLNumber。其他扩展可用，但默认禁用。这些可以被启用和修改。有关可用的 CRL 扩展的更多信息，请参阅 ???。

要配置 CRL 扩展，请执行以下操作：

1. 打开 CA 控制台。

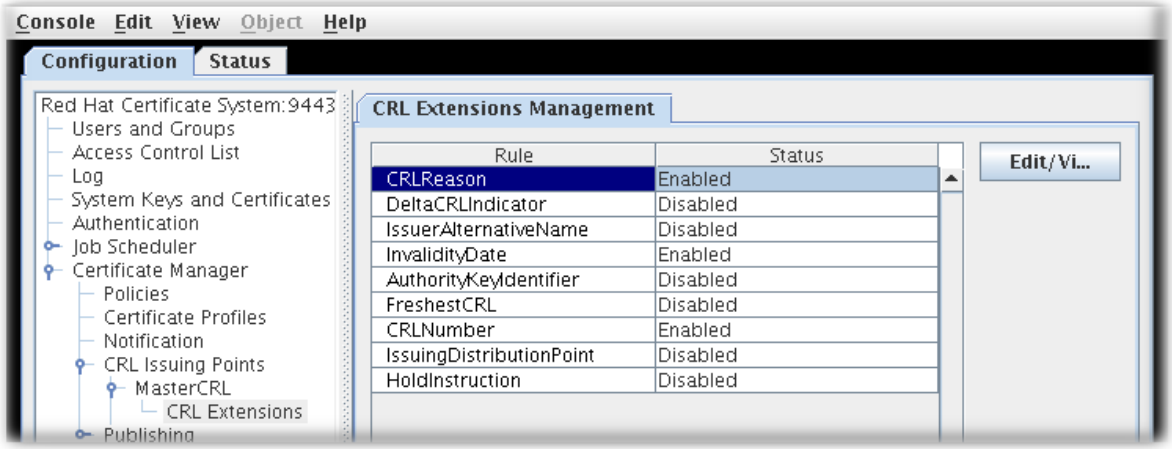
```
pkiconsole https://server.example.com:8443/ca
```

2. 在导航树中，选择 Certificate Manager，然后选择 CRL Issuing Points。

3. 选择 Issuing Points 条目下的发布点名称，然后选择发布点下的 CRL 扩展 条目。

   右侧窗格显示 CRL Extensions Management 选项卡，它列出了配置的扩展。

图 7.5. CRL 扩展

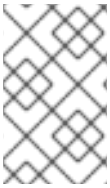


4. 若要修改规则，请选择它，然后单击 Edit/View。

5. 大多数扩展有两个选项，启用它们并设置它们是否至关重要。有些需要更多信息。提供所有必需的值。有关每个扩展以及这些扩展的参数的完整信息，请参阅 ???。

6. 点确定。

7. 点 Refresh 查看所有规则的更新状态。



注意

pkiconsole 已被弃用。

7.3.4. 将 CA 设置为使用不同的证书来签名 CRL

有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅 Red Hat Certificate System 规划、安装和部署指南中的设置 CA 来使用不同的证书签名 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，因为证书被撤销并保留在内存中。然后，可以使用此撤销信息从内存中更新 CRL。绕过从内部数据库生成 CRL 所需的数据库搜索可显著提高性能。

注意

由于性能增强从缓存生成 CRL，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 cache 测试 参数。

7.3.5.1. 在控制台中从缓存配置 CRL 生成

注意

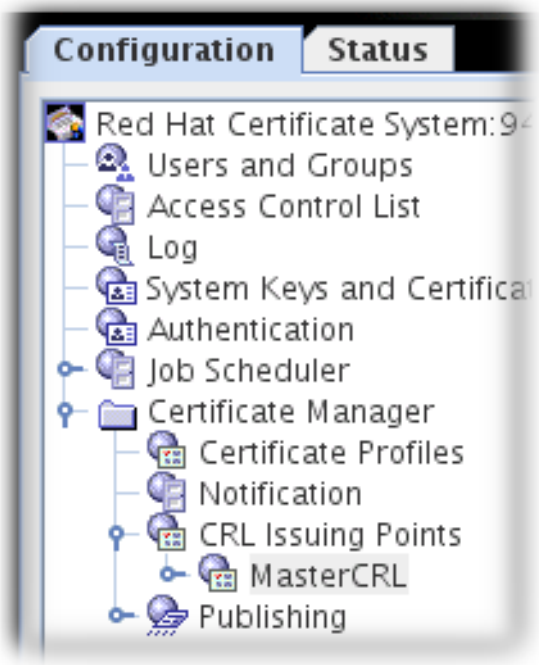
pkiconsole 已被弃用。

1. 打开控制台。

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

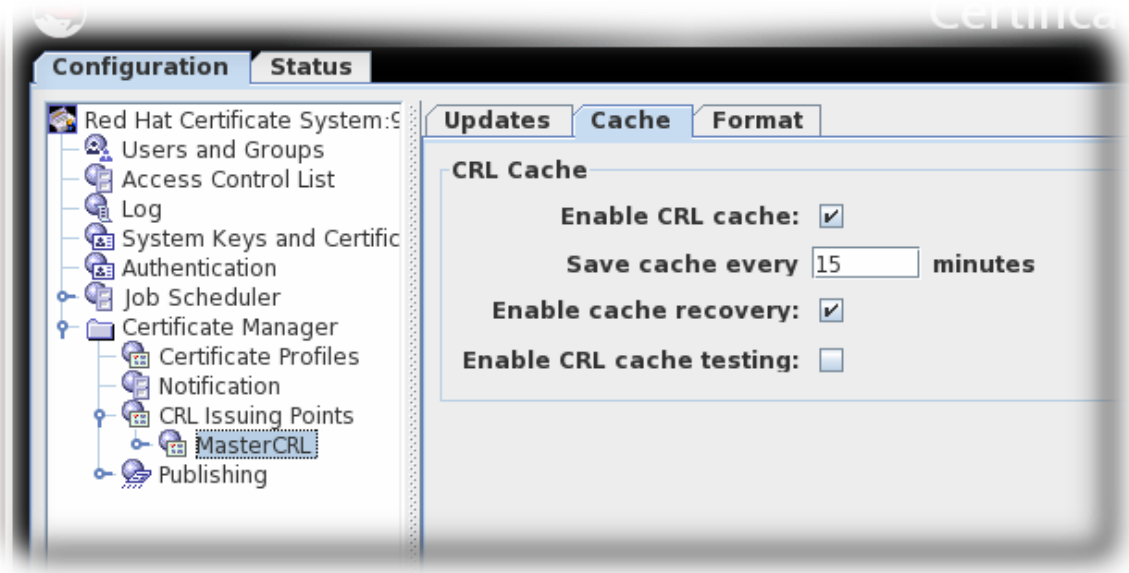
2. 在 Configuration 选项卡中，展开 Certificate Manager 文件夹和 CRL 颁发点 子文件夹。

3. 选择 MasterCRL 节点。



4.

*选择 **Enable CRL cache**。*



5.

保存更改。

7.3.5.2. 在 CS.cfg 中配置来自 Cache 的 CRL 生成

*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** 规划、安装和部署指南中的 **从 CS.cfg 中的配置 CRL 生成** 部分。*

7.4. 设置 FULL 和 DELTA CRL 计划

CRL 定期生成。在 第 7.3.2 节 “为每个颁发点配置 CRL” 中的配置中设定该周期。

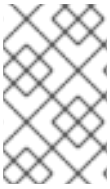
CRL 根据基于时间的调度进行。当证书被撤销、一天的特定时间或每这个分钟一次进行一次时，可以发布 CRL。

基于时间的 CRL 生成调度适用于生成的每个 CRL。CRL 有两种类型，即完全 CRL 和 delta CRL。完整的 CRL 具有每个撤销的证书的记录，而 delta CRL 则仅包含生成自最后一个 CRL （增量或完整）后撤销的证书。

默认情况下，在调度中的每个指定间隔都会生成完整的 CRL。通过生成内部 delta CRLs，有可能造成生成完整 CRL 之间的时间。生成间隔在 CRL 模式中配置，它会设置生成 delta 和 full CRL 的方案。

如果间隔设置为 3，则第一个 CRL 生成的是 full 和 delta CRL，则下一个两代更新仅是 delta CRL，然后第四个间隔是 full 和 delta CRL。换句话说，每个第三个间隔都具有完整的 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 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

注意

pkiconsole 已被弃用。

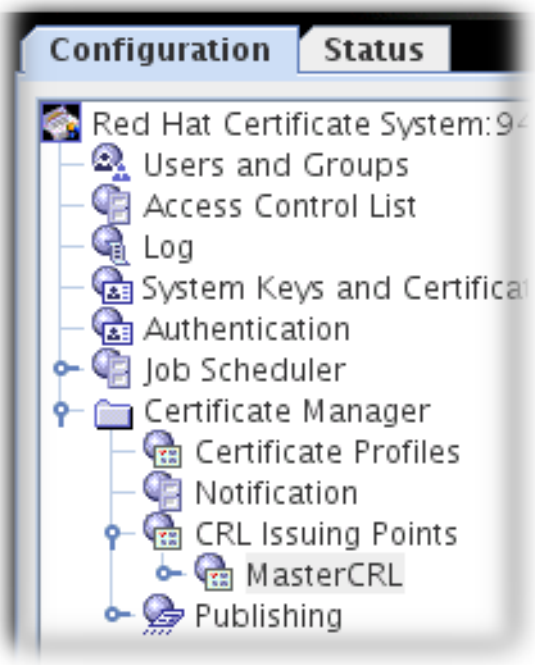
1. 打开控制台。

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

2. 在 Configuration 选项卡中，展开 Certificate Manager 文件夹和 CRL 颁发点 子文件夹。

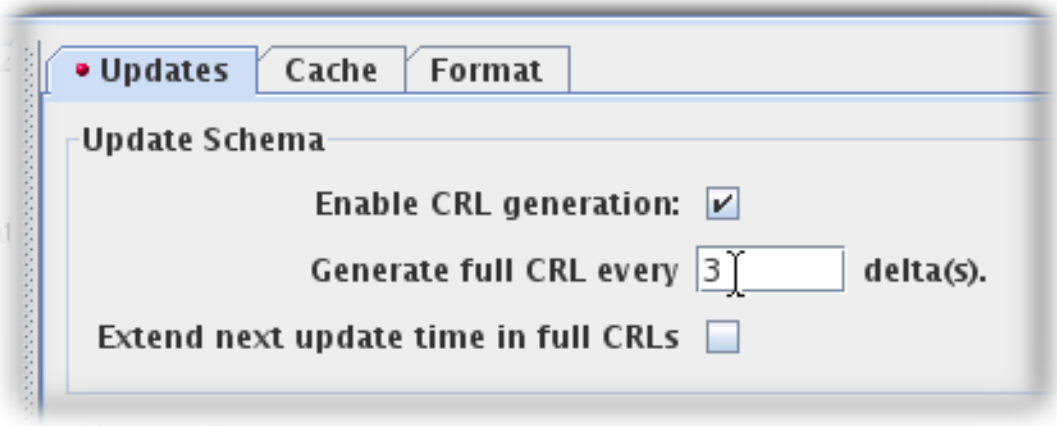
3.

*选择 **MasterCRL** 节点。*



4.

在 Generate full CRL (s) 字段中输入所需的间隔。



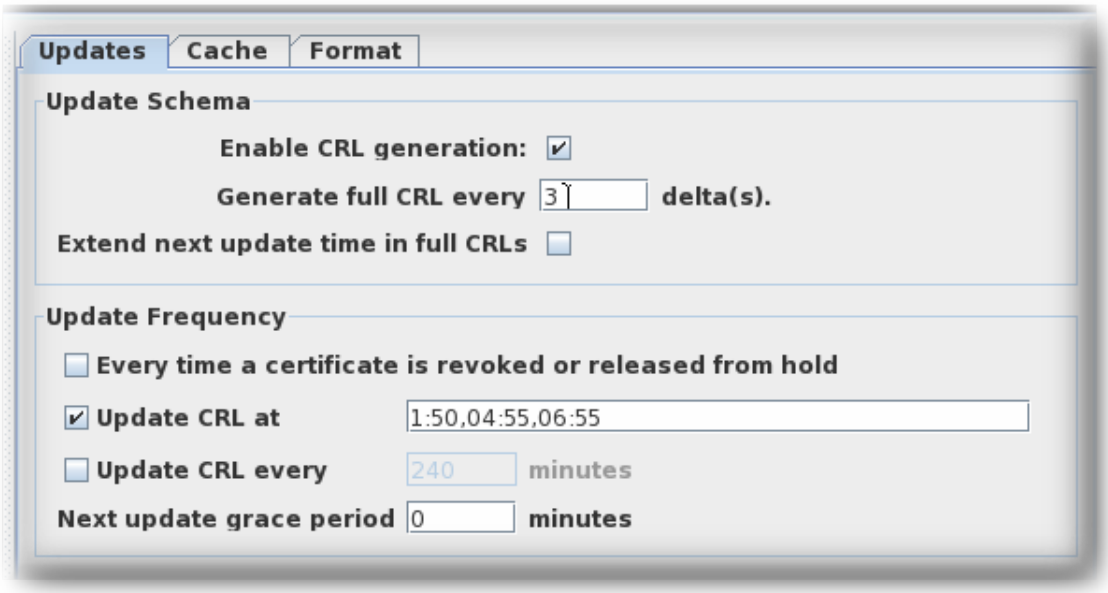
5.

通过指定证书撤销的 occasion、cyclical 间隔或设置更新的时间来设置更新频率：

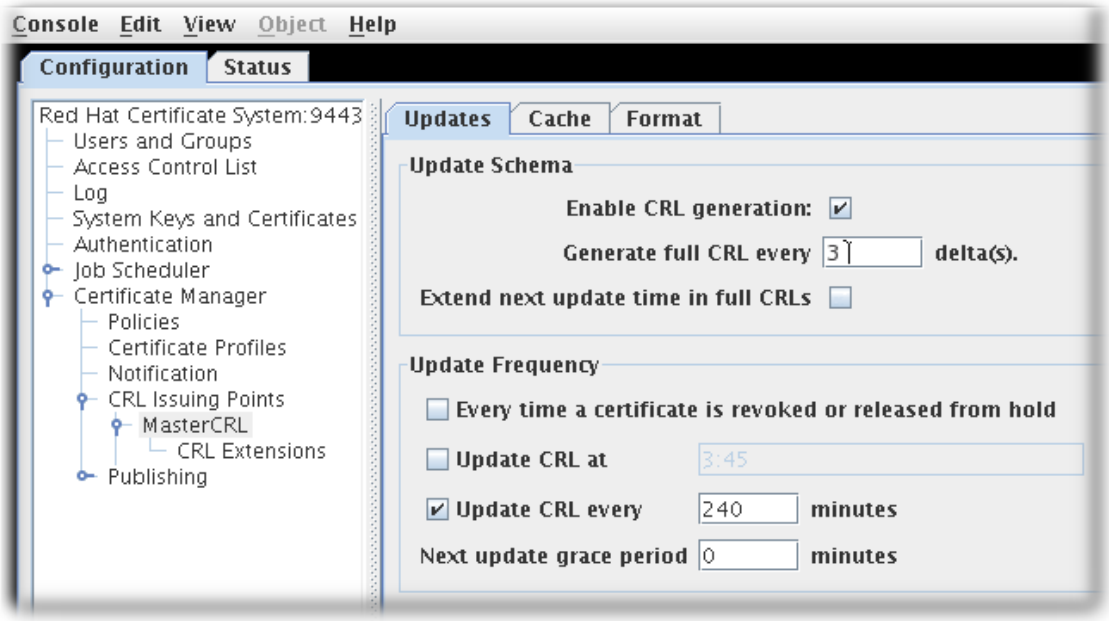
- *选择 **Update CRL，每次撤销或从暂停复选框中释放** 证书时。**每次从 hold** 选项撤销或发布证书时，**Update CRL 都需要填写两个 Grace period** 设置。这是一个已知问题，程序错误在 **Red Hat Bugzilla** 中被跟踪。*

- *选择 **Update CRL，每次撤销或从暂停复选框中释放** 证书时。*

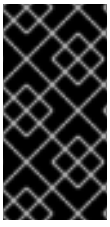
- *选择 **Update CRL at** 复选框，并输入用逗号分开的特定时间，如 **01:50,04:55,06:55**。*



- *选择 **Update CRL every** 复选框并输入所需的间隔，如 **240**。*



6. *保存更改。*



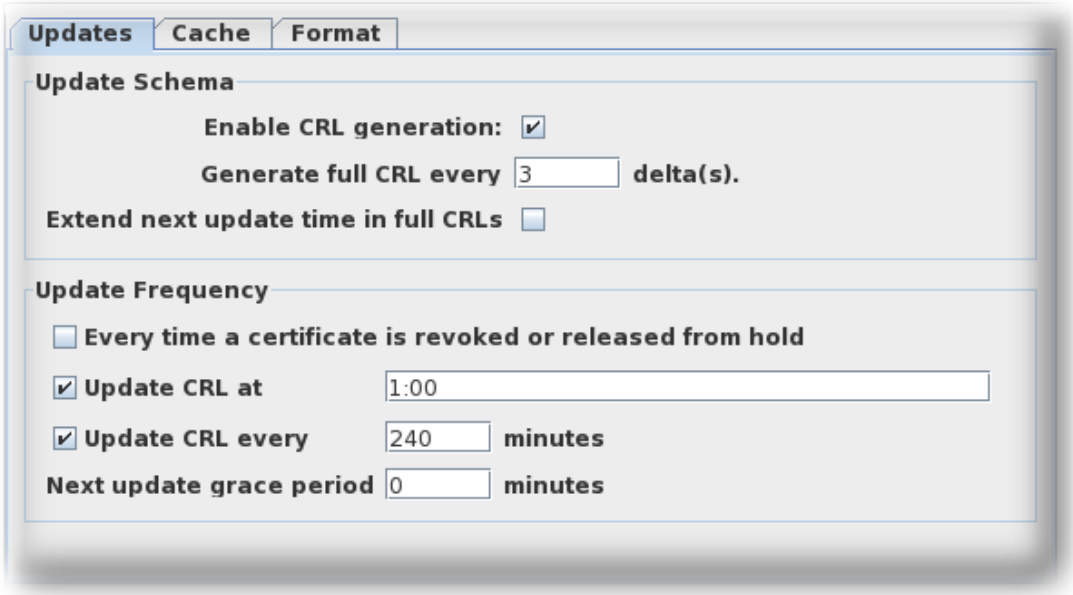
重要

***每次从 hold** 选项撤销或发布证书时，**Update CRL** 都需要填写两个 **宽限期** 设置。这是一个已知问题，程序错误在 **Red Hat Bugzilla** 中被跟踪。*

注意

根据间隔更新 CRL 时，可能会发生调度偏移。通常，因为手动更新和 CA 重启，会进行偏移。

要防止调度偏移，请选中 Update CRL at 复选框并输入值。间隔更新将每 24 小时的值 重新同步 Update CRL。



根据间隔 更新 CRL 时，只能接受一个 Update CRL。

7.4.2. 在 CS.cfg 中为 CRL 配置 Update Intervals

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 规划、安装和部署指南中的 为 CRL 配置 Update Intervals 部分。

7.4.3. 多次配置 CRL 生成计划

默认情况下，CRL 生成计划覆盖 24 小时。另外，当默认启用 full 和 delta CRLs 时，会以特定间隔或所有 delta CRLs 代替每个第三个更新。

要在多个天数内设置 CRL 生成调度，时间列表使用逗号分隔同一天内的时间，一个分号来取消限制天数：

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00;02:00,05:00,17:00
```

本例更新了位于 01:00, 03:00, 和 18:00 的时间表之日的 CRL，并在一天中的 02:00, 05:00, 和 17:00 调度中更新。在 3 天开始周期。

注意

分号表示一天。以分号开始列表会导致生成 CRL 的初始天。同样，以分号结尾的列表也会在没有生成 CRL 的调度中添加最后一天。两个分号一起会导致一天没有 CRL 生成。

要设置独立于 delta 更新的完整 CRL 更新，列表接受带有星号的时间值，以指示何时发生完整的 CRL 更新：

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00,*23:00;02:00,05:00,21:00,*23:30
```

本例每天在 01:00、03:00 和 18:00 时生成 delta CRL 更新，其完整和 delta CRL 更新于 23:00。在第二天，delta CRL 在 02:00、05:00 和 21:00 中更新，其完整和 delta CRL 更新为 23:30。在第 3 天，周期会再次开始。

注意

分号和星号语法可在控制台中和手动编辑 CS.cfg 文件时工作。

7.5. 启用撤销检查

吊销检查意味着证书系统子系统验证证书是否有效且在代理或管理员尝试访问实例的安全接口时没有被撤销。这利用本地 OCSP 服务(CA 的内部 OCSP 服务或单独的 OCSP 响应器)检查证书的撤销状态。

第 7.6 节 “使用在线证书状态协议(OCSP)恢复器” 中涵盖了 OCSP 配置。

请参阅 红帽证书系统规划、安装和部署 指南中的在 CA 上启用自动撤销检查。

请参阅 红帽证书系统规划、安装和部署指南中的 为子系统启用证书撤销检查。

7.6. 使用在线证书状态协议(OCSP)恢复器

7.6.1. 设置 OCSP Responder

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 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并信任。但是，如果选择了非安全域 CA，则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 OCSP Manager 所属的安全域中的每个 CA，而不是它的每个 CA 都会被 OCSP Manager 自动信任。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，但不能手动信任证书链。

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 服务处理的每个 CA 中设置发布规则(第 9 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读取，则不需要此项。

3. 证书配置文件必须配置为包含授权信息访问扩展，指向证书管理器侦听 OCSP 服务请求的位置(第 7.6.4 节 “启用证书管理器的内部 OCSP 服务”)。

4. 配置 OCSP Responder。

   - 配置撤销信息存储(第 7.6.2.2 节 “配置撤销信息存储：内部数据库” 和 第 7.6.2.3 节 “配置撤销信息存储：LDAP 目录”)。

   - 确定每个发布的证书管理器到 OCSP 响应器(第 7.6.2 节 “识别 CA 到 OCSP Responder”)。

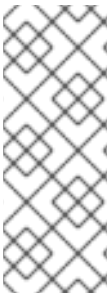
   - 如有必要，为签署 OCSP 签名证书(第 17.7 节 “更改 CA 证书的信任设置”)的 CA 配置信任设置。

5. 配置这两个子系统后重新启动这两个子系统。

6. 验证 CA 是否已正确连接到 OCSP 响应程序(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. 识别 CA 到 OCSP Responder

在将 CA 配置为向在线证书状态管理器发布 CRL 之前，必须在在线证书状态管理器的内部数据库中将 CA 识别为在线证书状态管理器。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针对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，则不需要额外步骤来配置在线证书状态管理器来识别 CA；CA 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，则必须在配置了在线证书状态管理器后手动将 CA 签名证书添加到证书数据库中。

不需要为 CA 导入证书链，后者将其 CRL 发布到在线证书状态管理器。OCSP 服务需要证书链的唯一时间是在发布 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完整的证书链。

但是，在线证书状态管理器必须具有为其证书数据库中的 CA 签名证书或单独的 CRL 签名证书的证书。OCSP 服务通过将 CRL 签名的证书与数据库中的证书进行比较来验证 CRL，而不是针对证书链。如果 root CA 及其子 CA 都向在线证书状态管理器发布 CRL，则在线证书状态管理器需要两个 CA 的 CA 签名证书。

要导入用于为 CA 发布的证书签名的 CA 或 CRL 签名证书，请执行以下操作：

1. 从 CA 的端到端页面获取证书管理器的 base-64 CA 签名证书。

2. 打开 Online Certificate Status Manager agent 页面。URL 格式为 https://hostname:SSLport/ocsp/agent/ocsp。

3. 在左侧帧中，单击 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（包括标头和页脚） 的文本区域中。

5. 要验证证书是否已成功添加，请在左侧范围内单击 List Certificate Authority。

生成的表单应该显示有关新 CA 的信息。此 Update、Next Update 和 Requests Served Since Startup 字段应该会显示 0 值(0)。

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已与在线证书状态管理器通信，请检查 此 更新和 下一步更新 字段，该字段应该使用与在线证书状态管理器的最后一个通信的适当时间戳进行更新。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。

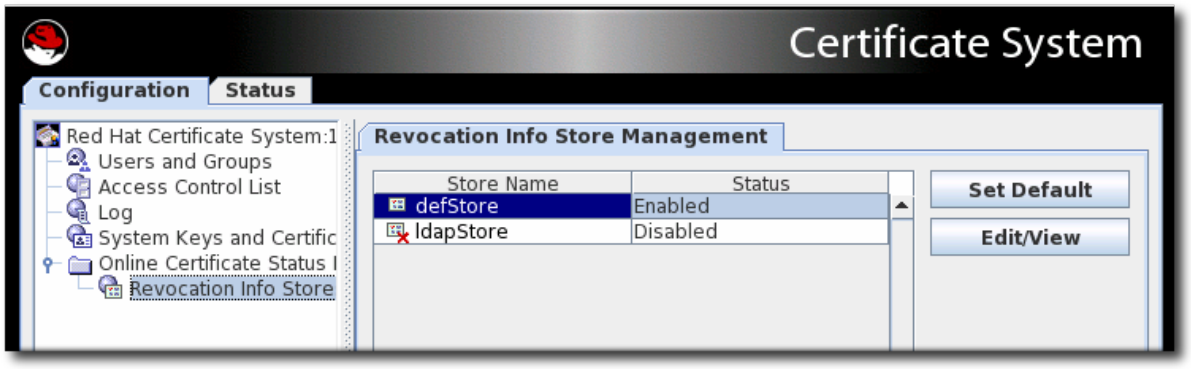
7.6.2.2. 配置撤销信息存储：内部数据库

在线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，并将它用作 CRL 存储以验证证书的撤销状态。要更改在线证书状态管理器用来在其内部数据库中存储 CRL 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。



右侧窗格中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，它会在其内部数据库中使用 CRL。

3. 选择 defStore，然后单击 Edit/View。

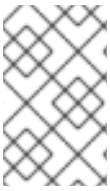
4. 编辑 defStore 值。

- notFoundAsGood.如果问题中的证书无法在任何 CRL 中找到，则将 OCSP 服务设置为返回 GOOD 的 OCSP 响应。如果没有选择此项，则响应为 UNKNOWN （当客户端遇到时）会导致错误消息。

- byName.OCSP Responder 只支持基本的响应类型，其中包括发出响应的 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 授权签名的证书主题名称将用

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将是 OCSP 响应的 ResponderID 字段。

- includeNextUpdate.包括下一个 CRL 更新时间的时间戳。



注意

pkiconsole 已被弃用。

7.6.2.3. 配置撤销信息存储：LDAP 目录

虽然 OCSP 管理器默认将 CA CRL 存储在其内部数据库中，但可以将其配置为改为使用发布到 LDAP 目录的 CRL。

重要

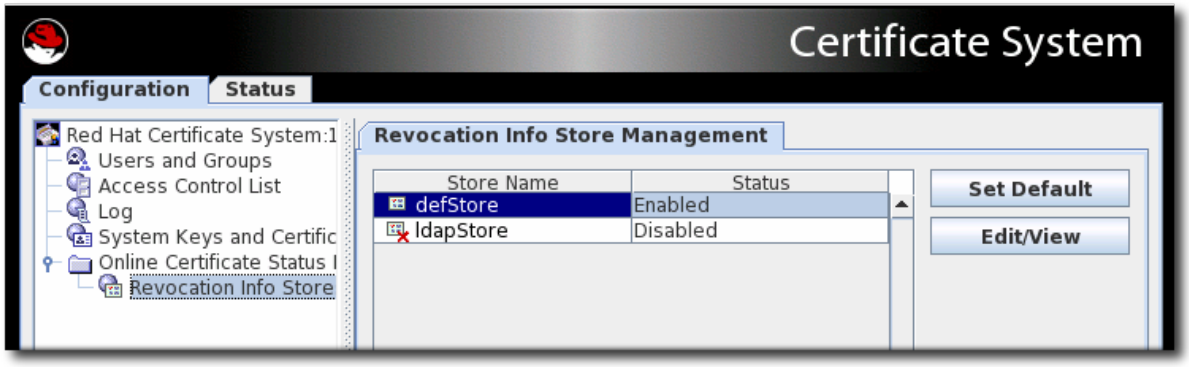
如果启用了 ldapStore 方法，则 OCSP 用户界面不会检查证书状态。

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。

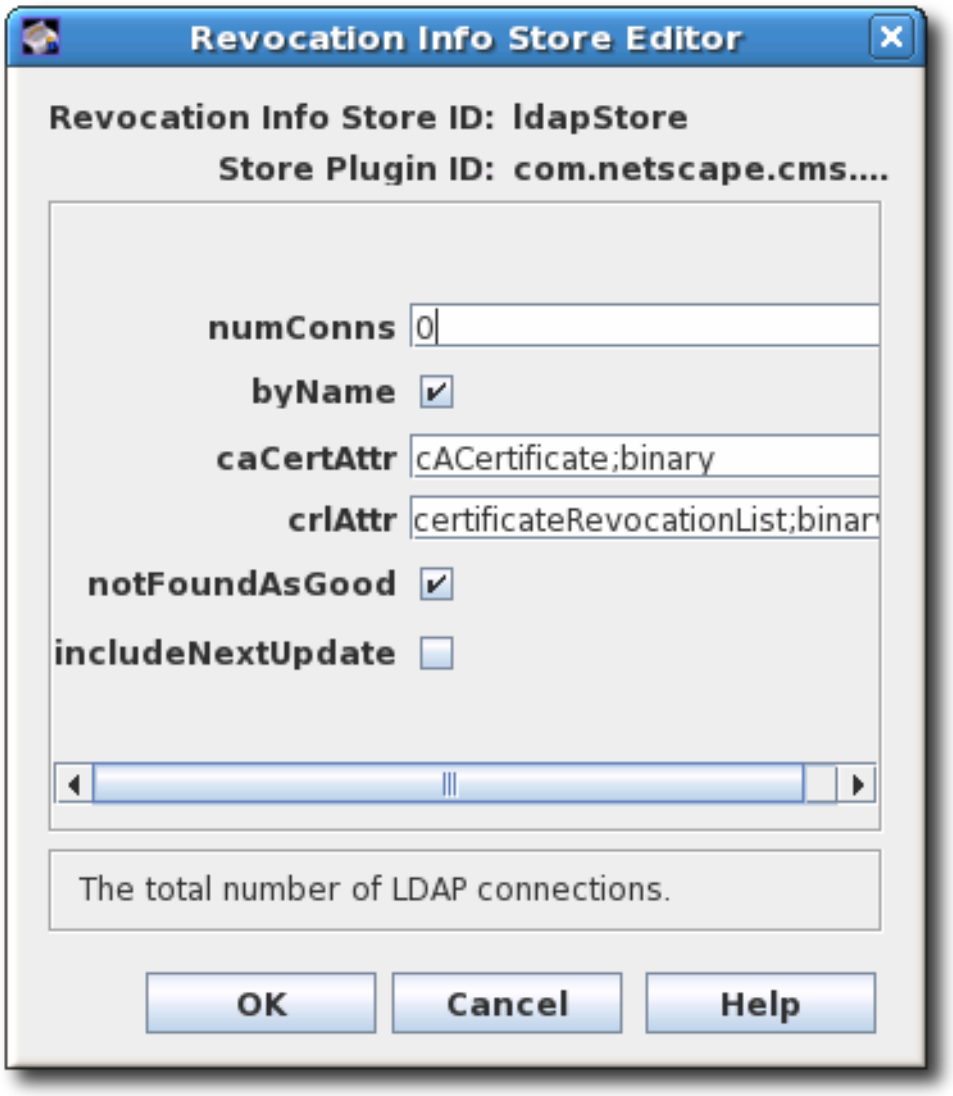


右侧窗格中显示在线证书状态管理器可以使用的两个存储库；默认情况下，它会在其内部数据库中使用 CRL。

3. *要在 LDAP 目录中使用 CRL，请点击 Set Default 来启用 ldapStore 选项。*

4. *选择 ldapStore，然后单击 Edit/View。*

5. *设置 ldapStore 参数。*



- ***numConns.OCSP 服务应检查的 LDAP 目录的总数。默认情况下，它被设置为 0。设置此值会显示对应的主机、端口、baseDN 和 refreshInSec 字段的数量。***

- ***host.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。连接刷新的频率。默认值为 86400 秒（每天）。***

- ***caCertAttr.保留默认值 cACertificate;binary，因为它是。它是证书管理器向其 CA 签名证书发布的属性。***

- crlAttr.保留默认值 certificateRevocationList;binary，因为它是。它是证书管理器向其发布 CRL 的属性。

- notFoundAsGood.如果问题中的证书无法在任何 CRL 中找到，则将 OCSP 服务设置为返回 GOOD 的 OCSP 响应。如果没有选择此项，则响应为 UNKNOWN （当客户端遇到时）会导致错误消息。

- byName.OCSP Responder 只支持基本的响应类型，其中包括发出响应的 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 授权签名的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将是 OCSP 响应的 ResponderID 字段。

- includeNextUpdate.在线证书状态管理器可以包含下一次 CRL 更新时间的时间戳。

注意

pkiconsole 已被弃用。

7.6.2.4. 测试 OCSP 服务设置

通过执行以下操作来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. 测试独立的在线证书状态管理器子系统。

   打开 Online Certificate Status Manager 代理服务页面，然后单击 List Certificate Authority 链接。

   该页面应当显示有关配置为向在线证书状态管理器发布 CRL 的证书管理器的信息。该页面还总结了在线证书状态管理器自上次开始以来的活动。

8. 吊销证书。

9. 在浏览器或客户端中验证证书。服务器应返回证书已被撤销。

10. 再次检查证书管理器的 OCSP-service 状态，以验证是否发生以下问题：

    - 浏览器向证书管理器发送 OCSP 查询。
    - 证书管理器向浏览器发送 OCSP 响应。
    - 使用该响应来验证证书并返回其状态的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以下问题：

    - 证书管理器将 CRL 发布到在线证书状态管理器。
    - 浏览器向在线证书状态管理器发送 OCSP 响应。

- 在线证书状态管理器向浏览器发送 OCSP 响应。

- 使用该响应来验证证书并返回其状态的浏览器无法验证证书。

7.6.3. 为 Bad Serial Numbers 设置响应

OCSP 响应器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，IADP 不会验证证书的其他信息。

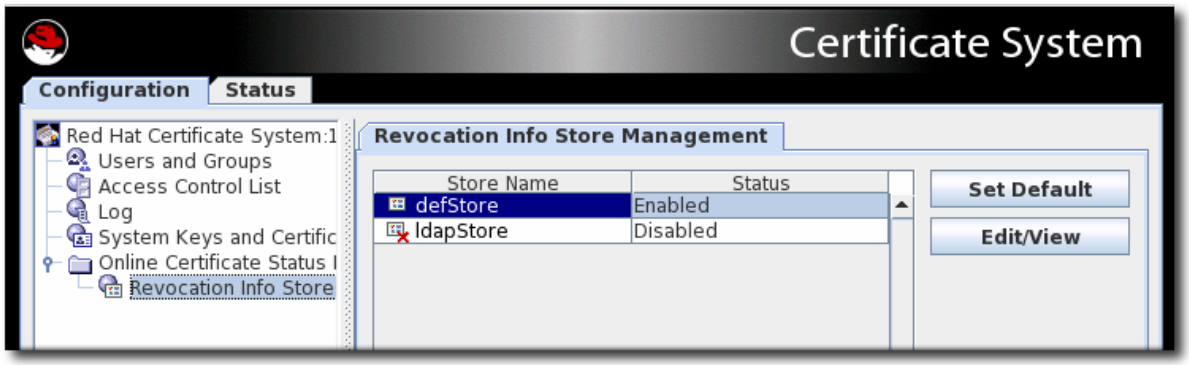
notFoundAsGood 参数设置 OCSP 如何使用无效序列号处理证书。此参数默认为启用，这意味着如果证书存在错误序列号但证书有效，则 OCSP 会返回证书的状态。

要获得基于错误的序列号以及撤销状态的 OCSP 检查和拒绝证书，请更改 notFoundAsGood 设置。在这种情况下，OCSP 返回 UNKNOWN 状态，并带有带有错误的序列号的证书。客户端将解释为错误，并相应地响应。

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，然后单击 Edit/View。

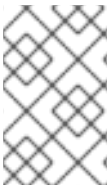
4.

编辑 notFoundAsGood 值。选择复选框意味着 OCSP 会返回 GOOD 值，即使证书的序列号不正确。取消选择复选框意味着 OCSP 发送一个 UNKNOWN 值，客户端可能会认为是错误。

5.

重启 OCSP Manager。

```
]# pki-server restart instance-name
```



注意

pkiconsole 已被弃用。

7.6.4. 启用证书管理器的内部 OCSP 服务

证书管理器具有内置的 OCSP 服务，可供 OCSP 兼容客户端用来直接查询证书管理器有关证书的撤销状态。安装证书管理器后，会发布 OCSP 签名证书，并默认启用 OCSP 服务。这个 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 服务检查证书管理器内部数据库中存储的证书状态，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 最终用户端口查询 OCSP 服务。当查询证书的撤销状态时，证书管理器会搜索其内部数据库的证书，检查其状态并响应客户端。由于证书管理器具有它发布的所有证书的实时状态，因此这种撤销检查的方法是最准确的。

每个 CA 的内置 OCSP 服务都会在安装时打开。但是，要使用此服务，CA 需要使用授权信息访问扩展发布证书。

1. 进入 CA 的端到端页面。例如：

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

2. 查找 CA 签名证书。

3. 在证书中查找授权信息访问扩展，并记录 Location URIName 值，如 **https://server.example.com:8443/ca/ocsp**。

4. 更新注册配置文件，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。有关编辑证书配置文件的详情，请参考 第 3.2 节 “设置证书配置文件”。

5. 重启 CA 实例。

```
]# pki-server restart instance-name
```

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 命令可与以下命令行选项一起使用：

表 7.1. 可用的 OCSPClient 选项

选项	描述
-d *数据库*	安全数据库位置（默认：当前目录）
-h *hostname*	OCSP 服务器主机名（默认为 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 nickname (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*	输出文件存储 DER 编码的 OCSP 响应的文件
-v,--verbose	以详细模式运行
--help	显示帮助信息

7.6.6. 使用 GET 方法提交 OCSP 请求

可以使用 GET 方法向在线证书状态管理器提交小于 255 字节的 OCSP 请求，如 RFC 6960 所述。通过 GET 提交 OCSP 请求：

1. 为证书生成一个 OCSP 请求，该请求正在查询。例如：

```
]# openssl ocsp -CAfile ca.pem -issuer issuer.pem -serial serial_number -reqout - | base64

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. 将 URL 粘贴到 Web 浏览器的地址栏中，以返回状态信息。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 使用浏览器可以解释的证书状态进行响应。可能的状态有 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. 为证书生成一个 OCSP 请求，该请求正在查询。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 来发送 OCSP 请求。

```
curl
https://server.example.com:8443/ocsp/ee/ocsp/MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4
cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy35o7xW5BMzM8FTvyTwCAQE= >
ocspresp.der
```

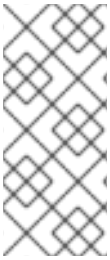
3. 使用 openssl 解析响应：

```
openssl ocsp -respin ocspresp.der -resp_text
```

对于由带有授权信息访问扩展的 7.1 CA 发布的证书使用 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 “为证书系统 7.1 和 Earlier 中发布的证书设置重定向” 所述。

7.6.7. 为证书系统 7.1 和 Earlier 中发布的证书设置重定向

OCSP 用户页面的位置在 URL 中指定，文件为 /ocsp/ee/ocsp/，在证书系统 10 或证书系统 8.1 中与证书系统 7.1 中的位置不同，这只是 /ocsp/。为了使 7.1 或更早的 CA 发布的证书以及授权信息访问扩展发送到 OCSP，请创建一个重定向以将请求转发到适当的 URL。



注意

只需要设置重定向，才能管理由带有授权信息访问扩展的 7.1 或更早的 CA 发布的证书。如果证书由更新的版本证书管理器或不包含授权信息访问扩展发布，则不需要此配置。

1. 停止 OCSP Responder。

```
]# pki-server stop instance-name
```

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. 更改为 OCSP Web 应用目录的 ROOT 文件夹中的 ROOT/WEB-INF/ 目录。例如：

```
]# cd /var/lib/pki-ocsp/webapps/ocsp/ROOT/WEB-INF/
```

4. 在 OCSP Web 应用目录的 ROOT 文件夹中创建并打开 lib/ 目录。

```
]# mkdir lib
]# cd lib/
```

5. 创建链接到 /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 ocsp/WEB-INF/ 目录中，编辑 web.xml 文件并添加 eeocspAddCRL 和 csadmin-wizard servlets 之间的行映射。***

```
  <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/ocsp/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。*

```
]# pki-server start instance-name
```

第 8 章 管理 PKI ACME RESPONDER

本章论述了如何管理 PKI ACME Responder。

有关如何设置 PKI ACME Responder 的详情，请参考 Red Hat Certificate System Planning、安装和部署指南中的 设置 PKI ACME Responder 章节。

8.1. 启用/禁用 ACME 服务

属于 Administrators 组的用户可以在 ACME 响应器中启用或禁用服务。用户可以通过基本身份验证或客户端证书身份验证进行身份验证。

- 要使用基本身份验证启用或禁用 ACME 服务，请指定用户名和密码：

```
$ pki -u <username> -p <password> acme-<enable/disable>
```

- 要使用客户端证书身份验证启用或禁用 ACME 服务，请指定证书 nickname 和 NSS 数据库密码：

```
$ pki -n <nickname> -c <password> acme-<enable/disable>
```

8.2. 检查 PKI ACME RESPONDER 的状态

- 要检查 ACME 响应器的状态，请运行以下命令：

```
$ pki acme-info
Status: Available
Terms of Service: https://www.example.com/acme/tos.pdf
Website: https://www.example.com
CAA Identities: example.com
External Account Required:false
```

如果服务被禁用，命令会显示以下结果：

```
$ pki acme-info
Status: Unavailable
```

注意

实际输出取决于 metadata.conf 配置文件中配置的内容。

部分 III. 管理 CA 服务的其他配置

第 9 章 发布证书和 CRL

Red Hat Certificate System 为证书颁发机构包括可自定义的发布框架，使证书颁发机构能够发布证书、证书撤销列表(CRL)和其他与证书相关的对象到任何受支持的软件仓库：LDAP 兼容目录、平面文件和在线验证机构。本章解释了如何配置证书管理器，将证书和 CRL 发布到文件、到目录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 响应器。

   根据要使用的位置数量，可以有一个发布者或多个发布者。位置可以通过证书和 CRL 或更严格的定义（如证书类型）分割。规则决定要发布的类型以及与发布者关联的位置。

2. 设置规则以确定将哪些证书发布到该位置。激活证书或 CRL 匹配的任何规则，因此同一证书可以发布到文件和 LDAP 目录，方法是匹配基于文件的规则并与基于目录的规则匹配。

   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。禁用所有不使用的规则。

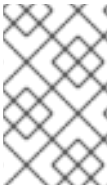
3. 配置 CRL。在发布 CRL 之前，必须配置 CRL。请参阅 第 7 章 吊销证书并颁发 CRL。

4. 在设置发布程序、映射程序和规则后启用发布。发布后，服务器会立即开始发布。如果没有完全配置发布程序、映射程序和规则，发布程序可能无法正常工作，或根本都无法正常工作。

9.1. 关于发布

证书系统能够向文件或 LDAP 目录发布证书，并将 CRL 发布到文件、LDAP 目录或 OCSP 响应器。

为获得更大的灵活性，可以发布特定类型的证书或 CRL，以单一格式或全部三种格式发布。例如，CA 证书只能发布到某个目录，不能发布到文件，用户可以将用户证书发布到文件和目录。



注意

OCSP 响应程序只提供有关 CRL 的信息；证书不会发布到 OCSP 响应程序。

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，不同类型的证书和不同类型的 CRL 可以发布到目录中的不同位置。例如，来自公司 West Coast 划分的用户的证书可以在目录的一个分支中发布，而 East Coast 划分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、更新或撤销证书或 CRL 时，都会调用发布系统。证书或 CRL 由规则评估，以查看是否与规则中设置的 type 和 predicate 匹配。type 指定对象是否为 CRL、CA 证书或其他证书。predicate 为要评估的对象类型设置更多条件。例如，可以指定用户证书，或者可以指定 West Coast 用户证书。要使用 predicates，需要在发布规则的 predicate 字段中输入一个值，并且需要包含在要匹配的证书或证书请求中相应的值（尽管有不同格式）。证书或证书请求中的值可以从证书中的信息（如证书的类型）派生，也可以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被视为匹配。例如，如果 CRL 设为类型，则所有 CRL 都与规则匹配。

匹配的每个规则都根据该规则中指定的方法和位置发布证书或 CRL。给定的证书或 CRL 不匹配规则、一条规则、多个规则或所有规则。发布系统会尝试匹配每个证书和针对所有规则发布的 CRL。

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，如果规则与签发给用户的所有证书匹配，并且该规则有一个发布者到位置 /etc/CS/certificates 中的文件，则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给 LDAP 属性 userCertificate;binary 属性的发布者，则证书将在用户条目的此属性中启用 LDAP 发布时发布到指定的目录中。

对于指定发布到文件的规则，会在证书或 CRL 在停滞目录中发布时创建一个新文件。

对于指定要发布到 LDAP 目录的规则，证书或 CRL 会在指定的属性中指定条目中发布。CA 使用任何后续证书或 CRL 覆盖任何发布的证书或 CRL 属性的值。简单地放置，任何已发布的现有证书或 CRL 都被下一个证书或 CRL 替代。

对于指定发布到在线证书状态管理器的规则，CRL 会发布到此管理器。证书没有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。映射程序用于决定要发布的条目。映射程序可以包含条目的确切 DN，一些变量可以关联证书获取的信息，以创建 DN，或者足够信息搜索条目中唯一属性或一组属性，以确定条目的正确 DN。

吊销证书时，服务器使用发布规则从 LDAP 目录或从文件系统中查找和移除对应的证书。

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，必须将服务器配置为运行适当的作业。详情请查看 第 13 章 设置自动化作业。

设置发布涉及配置发布程序、映射程序和规则。

9.1.1. publishers

publishers 指定发布证书和 CRL 的位置。在发布到文件时，发布者指定文件系统发布目录。在发布到 LDAP 目录时，发布者指定存储证书或 CRL 的目录的属性；映射程序用于确定条目的 DN。对于每个 DN，会为生成 DN 设置不同的公式。启用 LDAP 发布时，指定 LDAP 目录的位置。将 CRL 发布到 OCSP 响应器时，发布者指定在线证书状态管理器的主机名和 URI。

9.1.2. Mappers

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为条目构建 DN。服务器包含证书的主题名称和证书请求的信息，并且需要知道如何使用此信息为该条目创建 DN。映射程序提供了一个公式，用于将可用信息转换为 DN 或目录中可以搜索的一些唯一信息，以获取条目的 DN。

9.1.3. 规则

文件、LDAP 和 OCSP 发布 的规则 告诉服务器是否发布证书或 CRL。首先，规则通过为规则设置 type 和 predicate 来定义要发布的内容、证书或 CRL 匹配特定特征。然后，规则通过与发布者关联，并使用映射器指定发布方法和位置。

规则可以像 PKI 部署的要求一样简单或复杂，并足够灵活以适应不同的场景。

9.1.4. 发布到文件

服务器可以将证书和 CRL 发布到平面文件，然后可以导入到任何存储库，如关系数据库。当服务器配置为发布证书和 CRL 到文件时，发布的文件为 DER 编码的二进制 Blob、base-64 编码的文本 Blob 或两者。

- 对于服务器问题的每个证书，它创建一个文件，其中包含 DER 编码或 base-64 编码格式的证书。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，带有序列号 1234 的 DER 编码证书的文件名是 cert-1234.der。

- 每次服务器生成 CRL 时，它都会创建一个文件，其中包含以 DER 编码的或 base-64 编码格式的新 CRL。根据格式，每个文件都命名为 issue_point_name- this_update.der 或 issue_point_name- this_update.b64。issue_point_name 标识发布 CRL 的 CRL 发布点，并且 this_update 指定从文件中包含的 CRL 的相关更新值生成的值。例如，DER 编码的 CRL 的文件名，其值为 this Update:phone 28 15:36:00 PST 2020，是 MasterCRL-20200128-153600.der。

9.1.5. OCSP 发布

证书系统 OCSP 服务有两种形式，即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。内部服务没有设置为发布，它使用存储在其内部数据库中的证书来确定证书的状态。在线证书状态管理器检查证书管理器发送给的 CRL。为发送 CRL 的每个位置设置发布者，并为每种版本的 CRL 发送一个规则。

有关 OCSP 服务的详情，请参考 第 7.6 节 “使用在线证书状态协议(OCSP)恢复器”。

9.1.6. LDAP 发布

在 LDAP 发布 中，服务器使用 LDAP 或 LDAPS 将证书、CRL 和其他与证书相关的对象发布到目录。它发布的目录的分支称为 发布目录。

- 对于服务器问题的每个证书，它会在用户条目的指定属性中以 DER 编码格式包含证书的 Blob。证书作为 DER 编码的二进制 blob 发布。

- 每次服务器生成 CRL 时，它都会在 CA 条目的指定属性中创建一个包含新 CRL 的 Blob。

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，应用程序可以通过 HTTP 检索证书和 CRL。支持通过 HTTP 检索证书和 CRL 可让一些浏览器自动从服务器接收常规更新的目录导入最新的 CRL。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，用户条目必须存在于 LDAP 目录中。

如果因为某种原因，服务器和发布目录不同步，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 9.12.2 节 “在目录中手动更新 CRL”。

9.2. 配置发布到文件

配置发布的一般流程涉及设置发布者，以将证书或 CRL 发布到特定位置。根据要使用的位置数量，可以有一个发布者或多个发布者。位置可以通过证书和 CRL 或更精细的定义（如证书类型）分割。规则决定要发布的类型以及与发布者关联的位置。

发布到文件，只需将 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者；发布者不会自动创建发布者。要将所有文件发布到单个位置，请创建一个发布者。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，如用户证书或对象类型的子集，如 West Coast 用户证书。

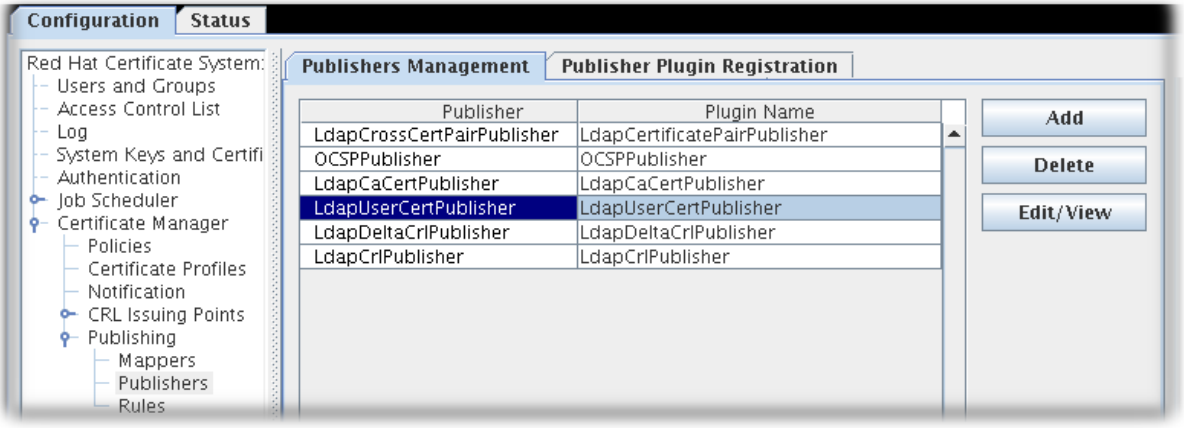
创建用于发布到文件的发布者：

1. 登录证书管理器控制台。

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

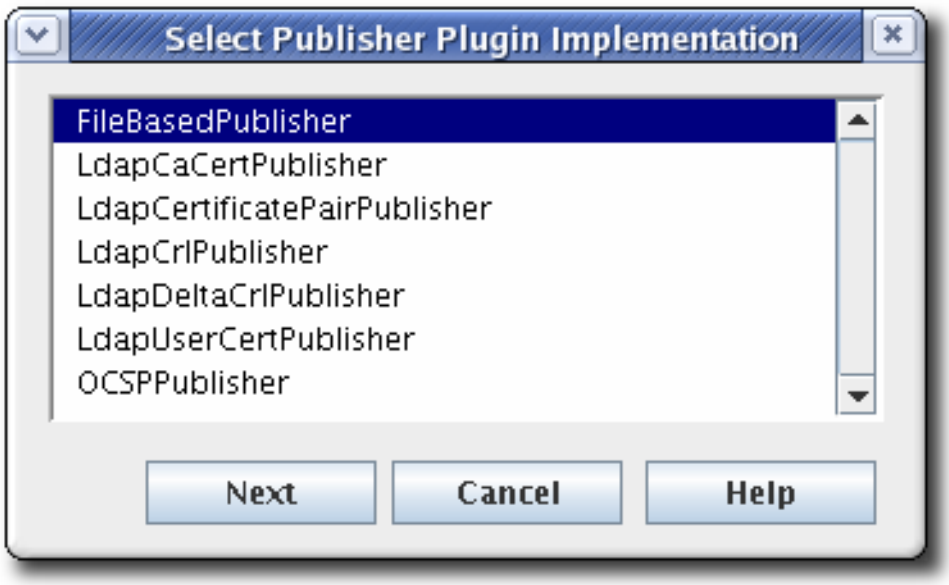
2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 Publishing，然后选择 Publishs。

Tailoring s Management 选项卡（列出配置的 publisher 实例）在右侧打开。



3.

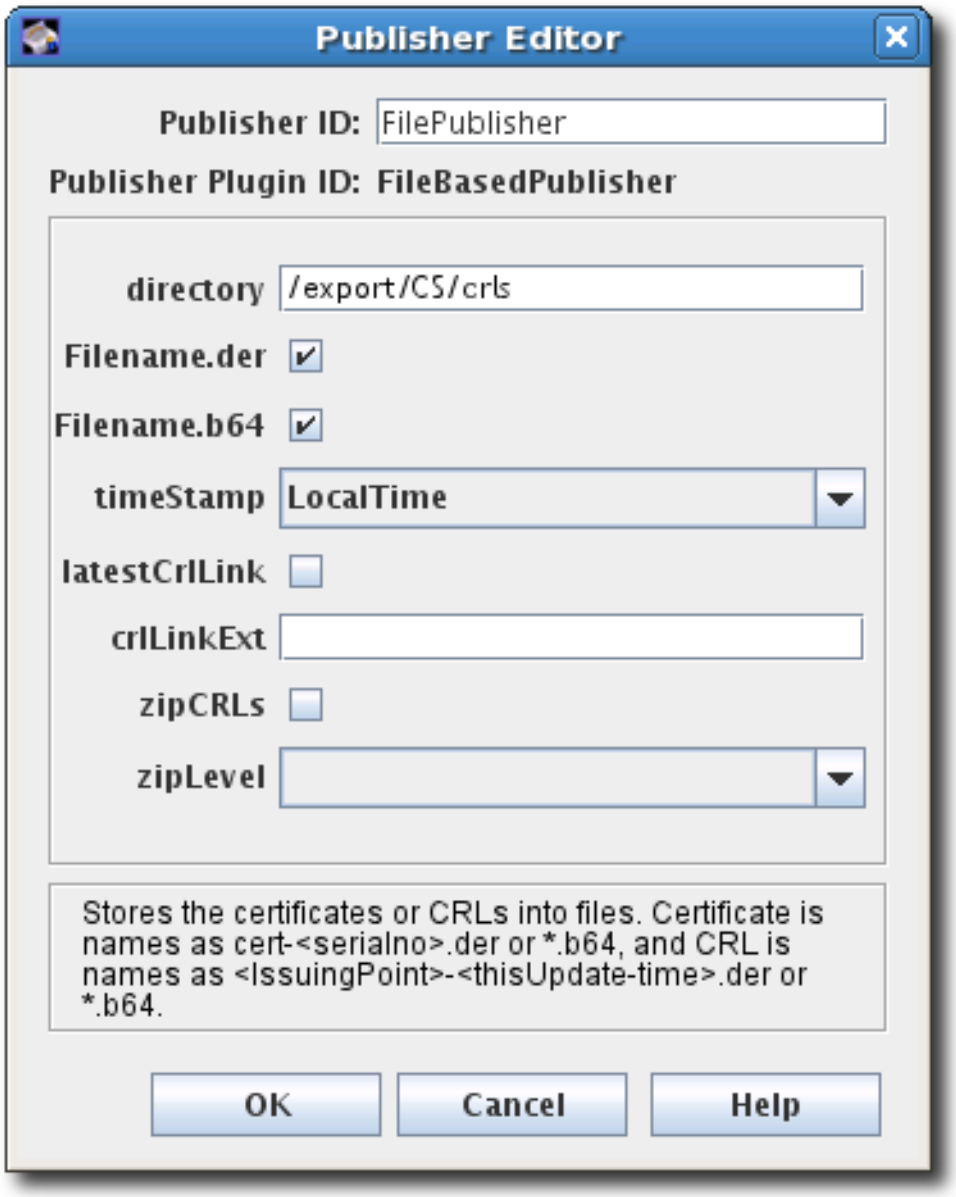
*单击 **Add** 以打开 **Select \":\" Plug-in Implementation** 窗口，该窗口列出了注册的发布者模块。*



4.

*选择 **FileBasedPublisher** 模块，然后打开编辑器窗口。*

*这是可让证书管理器向文件发布证书和 **CRL** 的模块。*



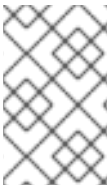
5. ***配置发布证书的信息：***

   - ***发布者 ID，它是一个没有空格的字母字符串，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，也可以相对于证书系统实例目录。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。发布的证书在其文件名中包含序列号，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，该链接假定要与扩展一起使用的 CRL 问题点的名称将在 crlLinkExt 字段中提供。

- 对于 CRL，是否压缩(zip) CRL 和要使用的压缩级别。

配置发布程序后，为发布的证书和 CRL 配置规则，如 第 9.5 节 “创建规则” 所述。



注意

pkiconsole 已被弃用。

9.3. 配置发布到 OCSP

配置发布的一般流程涉及设置发布者，以将证书或 CRL 发布到特定位置。根据要使用的位置数量，可以有一个发布者或多个发布者。位置可以通过证书和 CRL 或更精细的定义（如证书类型）分割。规则决定要发布的类型以及与发布者关联的位置。

发布到 OCSP 管理器是一种将 CRL 发布到特定位置以进行客户端验证的方法。

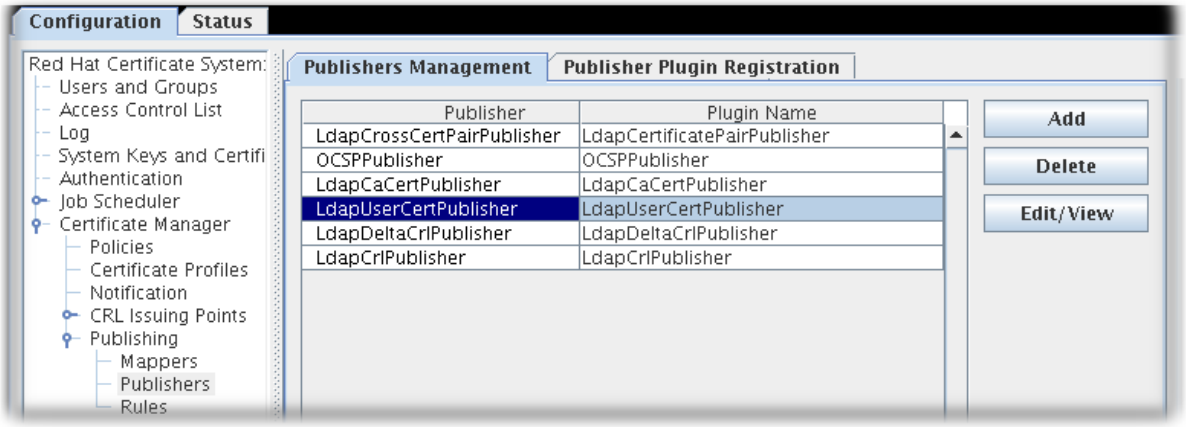
必须为每个发布位置创建和配置发布程序；发布到 OCSP 响应者不会自动创建发布者。创建一个发布者，将所有内容发布到单个位置，或为每个要发布 CRL 的位置创建一个发布者。每个位置都可以包含不同类型的 CRL。

9.3.1. 启用使用客户端身份验证发布到 OCSP

1. 登录证书管理器控制台。

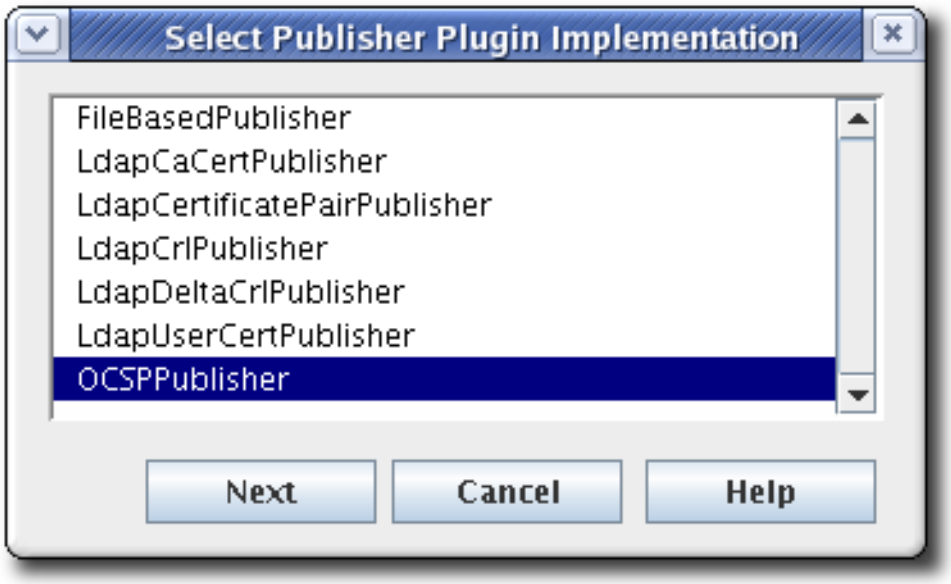
```
pkiconsole https://server.example.com:8443/ca
```

2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 Publishing，然后选择 Publishs。



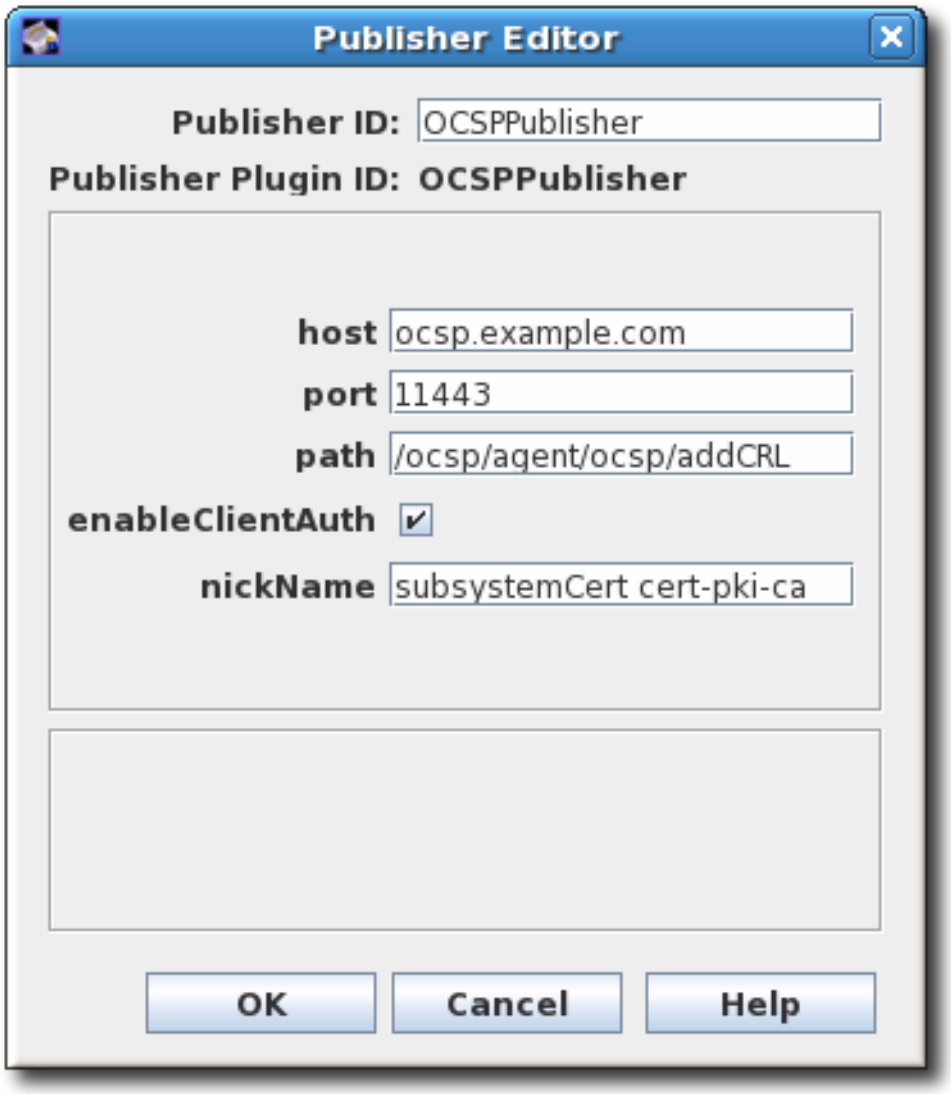
3.

*单击 **Add 以打开 Select \":\" Plug-in Implementation 窗口，该窗口列出了注册的发布者模块。***



4.

*选择 **OCSPPublisher 模块，然后打开编辑器窗口。这是发布者模块，使证书管理器能够向在线证书状态管理器发布 CRL。***



- *发布者 ID 必须是没有空格的字母字符串，如 PublishCertsToOCSP。*
- *主机 可以是完全限定域名，如 ocspResponder.example.com，也可以是 IPv4 或 IPv6 地址。*
- *默认路径是将 CRL 发送到的目录，如 /ocsp/agent/ocsp/addCRL。*
- *如果使用客户端身份验证（选中enableClientAuth），则 nickname 字段提供了用于身份验证的证书的 nickname 字段。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 中为 CA 创建用户条目。用户用于在发送新的 CRL 时向 OCSP 进行身份验证。需要两方面：*

- 在 CA 服务器后命名 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。

第 15.3.2.1 节 “创建用户” 中涵盖了设置子系统用户。

配置发布程序后，为发布的证书和 CRL 配置规则，如 第 9.5 节 “创建规则” 所述。

注意

pkiconsole 已被弃用。

9.4. 配置发布到 LDAP 目录

配置发布的一般流程涉及设置发布者，以将证书或 CRL 发布到特定位置。根据要使用的位置数量，可以有一个发布者或多个发布者。位置可以通过证书和 CRL 或更精细的定义（如证书类型）分割。规则决定要发布的类型以及与发布者关联的位置。

配置 LDAP 发布与其他发布过程类似，以及配置目录的额外步骤：

1. 配置要发布证书的目录服务器。某些属性必须添加到条目中，而且必须配置绑定身份和身份验证方法。

2. 为公布的每种对象配置发布者：CA 证书、跨对证书、CRL 和用户证书。publisher 声明用于存储对象的属性。默认设置的属性是 X.500 标准属性，用于存储每种对象类型。此属性可以在发布程序中更改，但通常不需要更改 LDAP 发布者。

3. 设置映射程序，以启用条目的 DN 从证书的主题名称中派生。这通常不需要为 CA 证书、CRL 和用户证书设置。对于一种证书，可以有多个映射程序。例如，这可用于发布来自位于目录树不同部分公司不同部分的两组用户的证书。为每个组创建一个映射程序，以指定树的不同分支。

有关设置映射程序的详情，请参考 第 9.4.3 节 “创建映射程序”。

4. 创建规则以将发布程序连接到映射程序，如 第 9.5 节 “创建规则” 所述。

5. 启用发布，如 第 9.6 节 “启用发布” 所述。

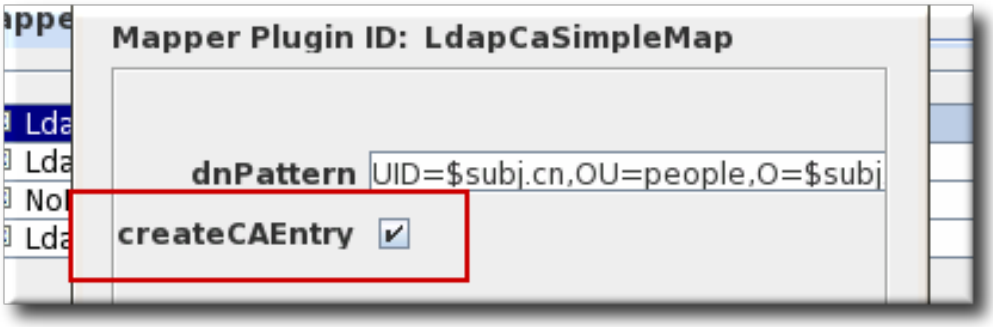
9.4.1. 配置 LDAP 目录

在发布证书和 CRL 之前，必须将目录服务器配置为使用发布系统。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，必须创建条目来代表 CRL。

1. 为 CA 设置条目。要使证书管理器发布其 CA 证书和 CRL，该目录必须包含 CA 的条目。

TIP

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。这个选项在 CA 和 CRL mapper 实例中设置，并默认启用。如果目录限制了证书管理器在目录中创建条目，请在这些映射器实例中关闭此选项，并在目录中手动为 CA 添加条目。



将 CA 条目添加到目录中时，根据 CA 的 DN 选择条目类型：

- 如果 CA 的 DN 从 cn 组件开始，请为 CA 创建新的 person 条目。选择其他类型的条目可能无法指定 cn 组件。

- 如果 CA 的 DN 从 ou 组件开始，请为 CA 创建新的 organizationalunit 条目。

条目不必位于 pkiCA 或 Certification Authority 对象类中。证书管理器将通过发布其 CA 的签名证书，将此条目转换为 pkiCA 或 Certification Authority 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 服务器使用的模式定义，任何对象类都可以接受。在某些情况下，两个对象类都可用于相同的 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. ***将正确的架构元素添加到 CA 和用户条目条目中。***

要使证书管理器向目录发布证书和 CRL，必须使用特定属性和对象类进行配置。

对象类型	模式	原因
端到端证书	userCertificate;binary (attribute)	这是证书管理器向其发布证书的属性。 这是一个多值属性，每个值都是一个 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，以允许将证书发布到其他对象类的目录条目中。证书管理器不会自动将此对象类添加到对应的目录服务器的 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)	这是证书管理器向其发布证书的属性。 证书管理器在服务器启动时将自己的 CA 证书发布到自己的 LDAP 目录条目。条目对应于证书的签发者名称。 这是 **pkiCA** 或 Certification **Authority 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，则证书管理器会将此对象类添加到 CA 的目录条目中。

对象类型	模式	原因
CRL	certificateRevocationList;binary (attribute)	这是证书管理器向发布 CRL 的属性。 证书管理器将 CRL 发布到自己的 LDAP 目录条目。条目对应于证书的签发者名称。 这是 **pkiCA** 或 Certification **Authority** 对象类的属性。属性的值是 DER 编码的二进制 X.509 CRL。CA 的条目必须已包含 CRL 的 **pkiCA** 或 Certification **Authority** 对象类，才能发布到该条目。
delta CRL	deltaRevocationList;binary (attribute)	这是证书管理器向发布 delta CRL 的属性。证书管理器将 delta CRL 发布到自己的 LDAP 目录条目，与完整 CRL 分开。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 certified **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，用于访问目录服务器。*

证书管理器用户必须具有目录的读写权限，才能向目录发布证书和 CRL，以便证书管理器可以使用与证书相关的信息以及带有 CA 证书和 CRL 相关信息的 CA 条目修改用户条目。

绑定 DN 条目可以是以下之一：

- *具有写入访问权限的现有 DN，如目录管理器。*
- *授予写入访问权限的新用户。条目可以被证书管理器的 DN 标识，如 cn=testCA, ou=Research Dept, o=Example 公司, st=California, c=US。*

注意

仔细考虑为此用户授予哪些特权。此用户可以通过为帐户创建 ACL 限制到目录的内容。有关授予对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. 设置目录身份验证方法，以向目录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 SSL （基于证书）的 SSL。

有关设置这些与服务器通信方法的说明，请参阅 Red Hat Directory Server 文档。

9.4.2. 配置 LDAP 站

证书管理器创建、配置和启用与 LDAP 发布关联的一组发布者。默认发布者（用于 CA 证书、用户证书、CRL 和跨修复证书）已符合存储证书和 CRL 的 X.500 标准属性，不需要更改。

表 9.1. LDAP 过程

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 CRLs 发布到 LDAP 目录。
LdapUserCertPublisher	将所有类型的最终用户证书发布到 LDAP 目录。
LdapCrossCertPairPublisher	将自签名证书发布到 LDAP 目录。

9.4.3. 创建映射程序

映射程序仅与 LDAP 发布一起使用。映射程序定义了证书的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派生条目的 DN，以便它能够决定要使用哪个条目。mapper 定义用户条目的 DN 和证书的主题名称或其他输入信息之间的关系，以便可以在目录中确定和找到条目的确切 DN。

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。默认映射程序列在 表 9.2 “默认

映射程序” 中。

表 9.2. 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。

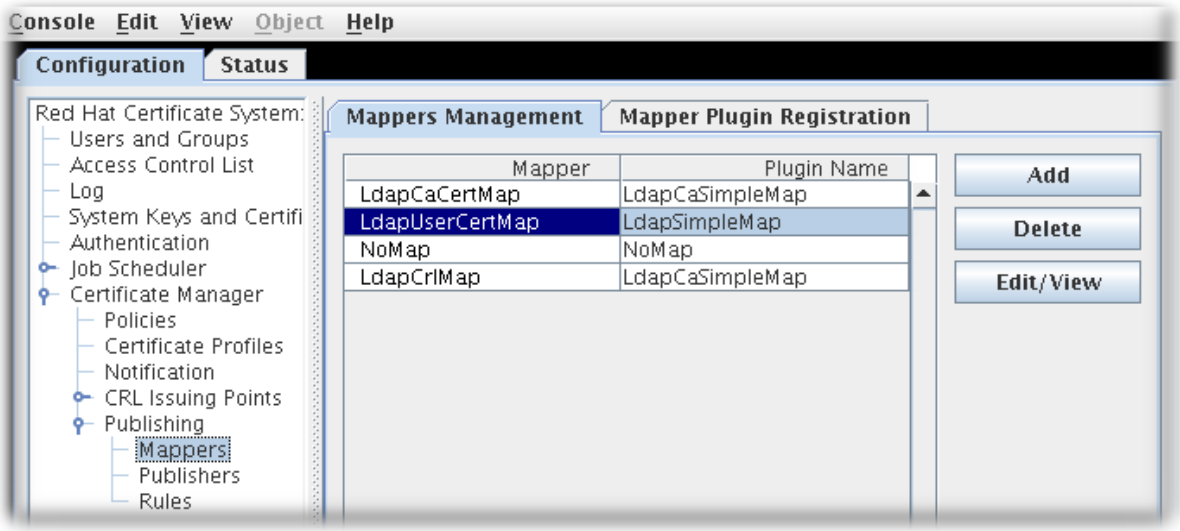
要使用默认映射程序，请通过指定 DN 模式并在目录中创建 CA 条目来配置每个宏。要使用其他映射程序，请创建并配置映射器实例。如需更多信息，请参阅 第 C.2 节 “映射程序插件模块 ”。

1. *登录证书管理器控制台。*

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

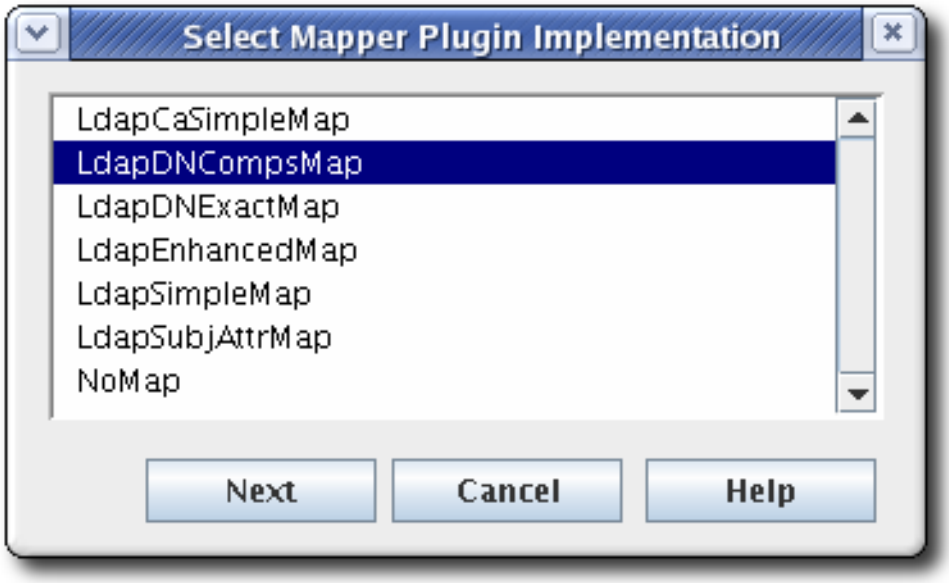
2. *在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 Publishing，然后选择 Mappers。*

Mappers Management 标签页（列出配置的映射程序）在右侧打开。

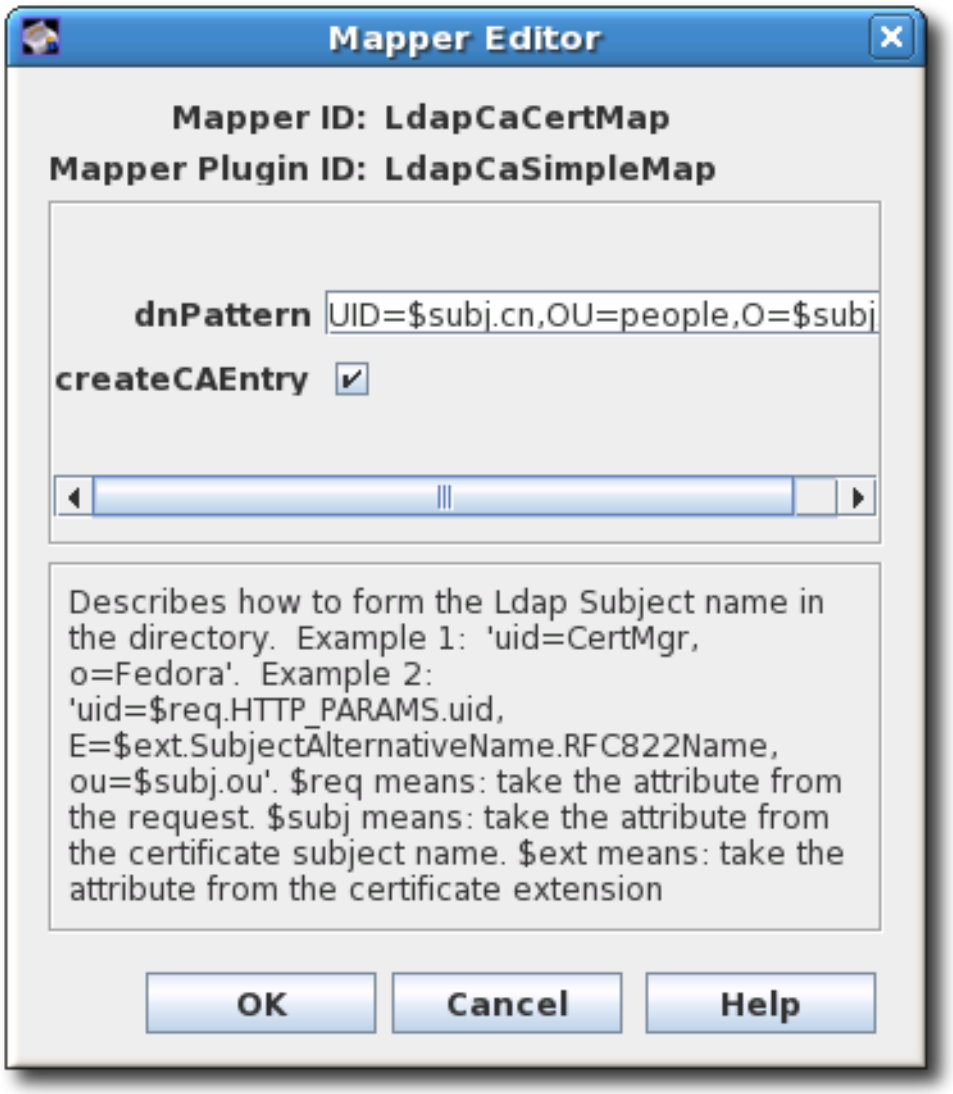


3. *若要创建新映射程序实例，请单击 Add。此时会打开 Select Mapper Plugin*

Implementation 窗口，它列出了注册的映射程序模块。选择一个模块并编辑它。有关这些模块的完整信息，请参考 第 C.2 节 “映射程序插件模块”。



4. *编辑 mapper 实例，然后单击 OK。*



有关每个映射程序的详情，请查看 第 C.2 节 “映射程序插件模块 ”。

注意

pkiconsole 已被弃用。

9.4.4. 完成配置：规则并启用

为 LDAP 发布程序配置映射程序后，为发布的证书和 CRL 配置规则，如 第 9.5 节 “创建规则” 所述。

配置完成后，启用发布，如 第 9.6 节 “启用发布” 所述。

9.5. 创建规则

规则决定了在哪些位置发布哪些证书对象。规则独立工作，而不是在 tandem 中工作。要发布的证书或 CRL 会根据每个规则进行匹配。激活它匹配的规则。这样，可以将同一证书或 CRL 发布到文件、在线证书状态管理器，并通过匹配基于文件的规则、OCSP 规则和匹配基于目录的规则来发送到 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。对于不同类型的证书或不同类型的 CRL，规则可以更加详细。

规则首先根据与对象的规则中设置的 type 和 predicate 匹配。发布匹配的对象由与规则关联的发布者和映射程序决定。

为证书管理器发布的每种证书创建规则。

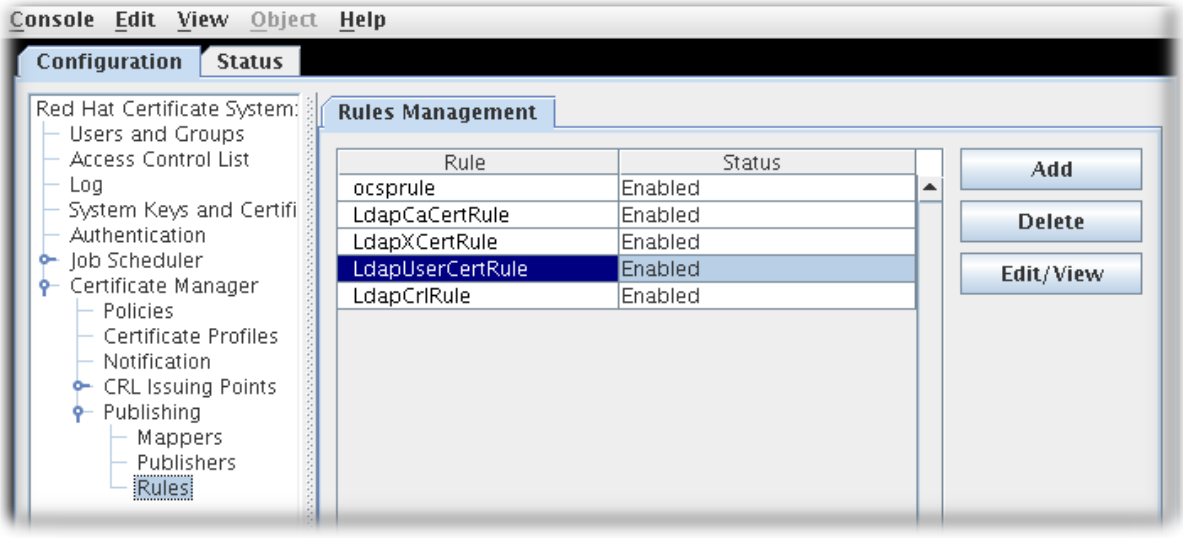
通过执行以下操作来修改发布规则：

1. 登录证书管理器控制台。

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

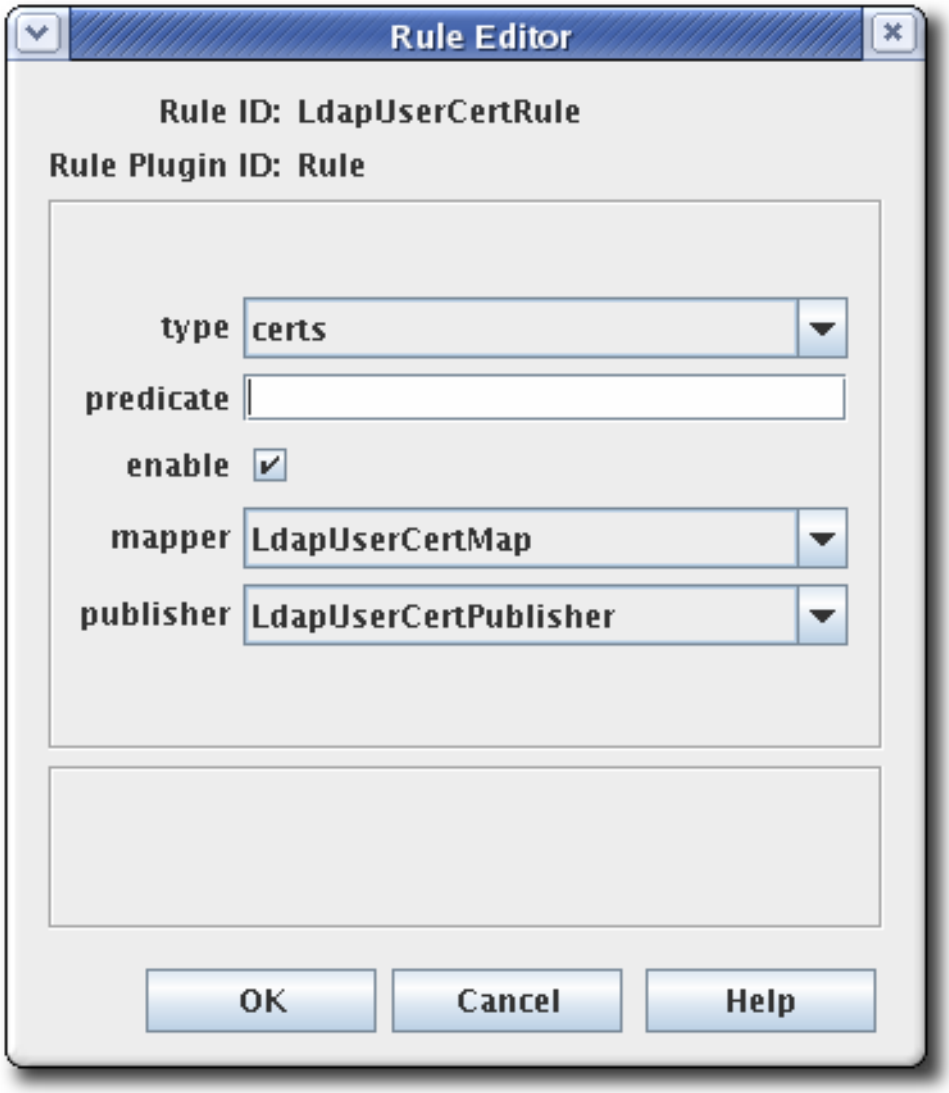
2. 在 Configuration 选项卡中，从左侧的导航树中选择 Certificate Manager。选择 Publishing，然后选择 Rules。

Rules Management 选项卡（列出配置的规则）在右侧打开。



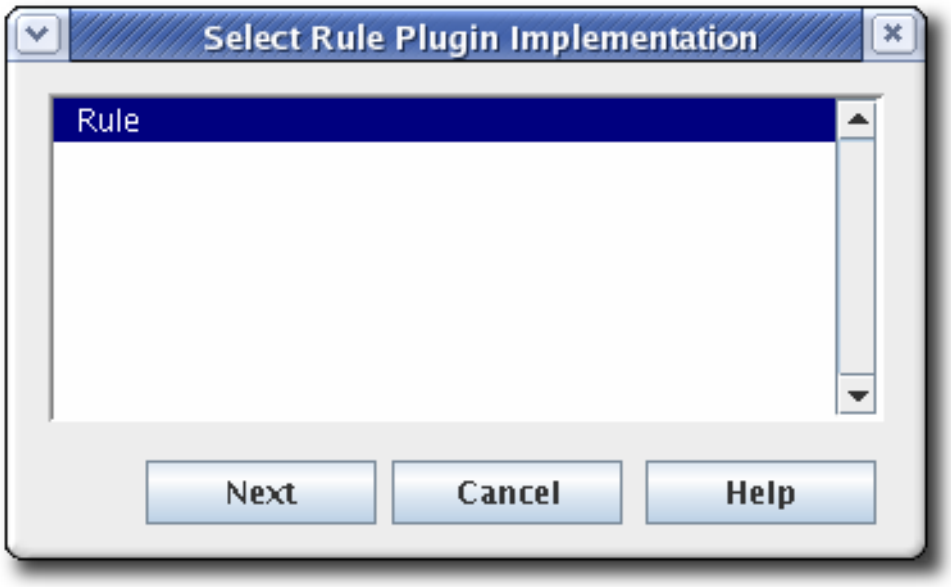
3.

若要编辑现有规则，请从列表中选择该规则，然后单击 **Edit**。*这将打开* **Rule Editor** *窗口。*



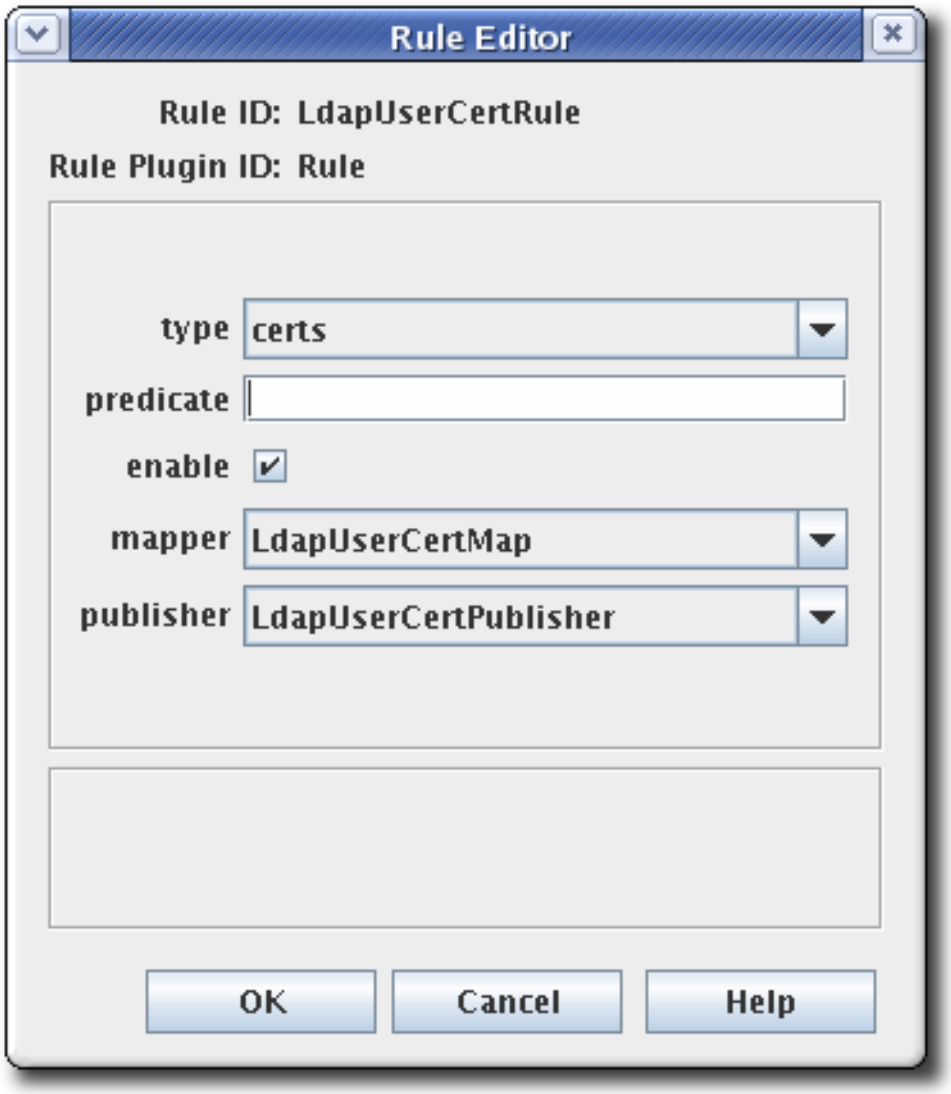
4.

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。这是唯一的默认模块。如果注册了任何自定义模块，它们也可用。*

5. *编辑规则。*



- *type.这是规则适用的证书类型。对于 CA 签名证书，值为 cacert。对于跨林信任，值为 xcert。对于所有其他证书类型，值为 certs。对于 CRL，指定 crl。*

- *predicate。这会为规则适用的证书或 CRL 发布点设置 predicate 值。CRL 发布点、delta CRL 和证书的 predicate 值列在 表 9.3 “predicate 表达式” 中。*

- *启用。*

- *映射器.发布到文件时不需要映射程序；仅 LDAP 发布需要它们。如果此规则与发布到 LDAP 目录的发布程序关联，请在此处选择适当的映射程序。对于所有其他形式的发布，请留空。*

- *发布者.设置要与规则关联的发布程序。*

表 9.3 “predicate 表达式” 列出可用于识别 CRL 发布点和 delta CRLs 和证书配置文件的 predicates。

表 9.3. predicate 表达式

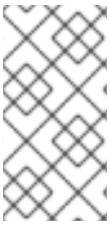
predicate Type	predicate
CRL 颁发点	**issuingPointId==***Issuing_Point_Instance_ID* **&& isDeltaCRL==[true|false]** 要仅发布 master CRL，请设置 **isDeltaCRL==false**。要仅发布 delta CRL，请设置 **isDeltaCRL==true**。要发布这两者，请为 master CRL 设置一条规则，并为 delta CRL 设置另一个规则。
证书配置文件	**profileId==***profile_name* 要根据用于发布它们的配置集发布证书，请将 **profileId==** 设置为配置集名称，如 **caServerCert**。

注意

pkiconsole 已被弃用。

9.6. 启用发布

发布只能为文件、只有 LDAP 或两者都启用。在设置发布程序、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。如果在启用前没有正确配置发布，发布可能会带来不必要的行为，或者可能会失败。



注意

配置 CRL。在发布 CRL 之前，必须配置 CRL。请参阅 第 7 章 吊销证书并颁发 CRL。

1. *登录证书管理器控制台。*

```
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 Publishing。*

右侧窗格显示发布到符合 LDAP 的目录的详细信息。

3. 若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 和 Enable Default LDAP Connection。

在 Destination 部分中，设置 Directory 服务器实例的信息。

- 主机名.如果为 SSL 客户端经过身份验证的通信配置了目录服务器，该名称必须与 Directory 服务器 SSL 服务器证书的主题 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。这是具有目录管理器权限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。仅针对发布系统实际需要写入的属性，可以创建另一个具有有限读写权限的 DN。

-

密码。这是 CA 用于绑定到发布证书或 CRL 的 LDAP 目录的密码。证书管理器将此密码保存到其 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 参数中的证书管理器 CS.cfg 文件中设置，可以对其进行编辑。

- 客户端证书.这会将证书管理器用于 SSL 客户端身份验证的证书设置为发布目录。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.证书管理器向目录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。

  如果将目录服务器配置为基本身份验证或在没有客户端身份验证的情况下 SSL 通信，请选择 基本身份验证 并为 Directory Manager DN 和密码指定值。

  如果为与客户端身份验证的 SSL 通信配置了目录服务器，请选择 SSL 客户端身份验证 和 Use SSL 通信 选项，并且识别证书管理器必须用于 SSL 客户端对目录进行身份验证的证书。

服务器尝试连接到目录服务器。如果信息不正确，服务器会显示错误消息。

注意

pkiconsole 已被弃用。

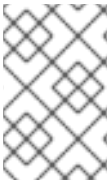
9.7. 启用发布队列

注册过程的一部分包括将发布的证书发布到任何目录或文件。这基本上，关闭初始证书请求。但是，向外部网络发布证书可能会显著减慢问题，这会使请求保持打开。

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）与请求和注册操作分开，后者使用单独的请求队列。请求队列会立即更新，以显示注册过程已完成，发布队列则以网络流量的速度发送信息。

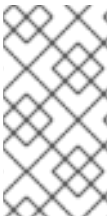
发布队列设置定义的、有限的线程数量，它们发布生成的证书，而不是为每个批准的证书打开新的线程。

发布队列默认为禁用。它可以在 CA 控制台中启用，以及启用发布。



注意

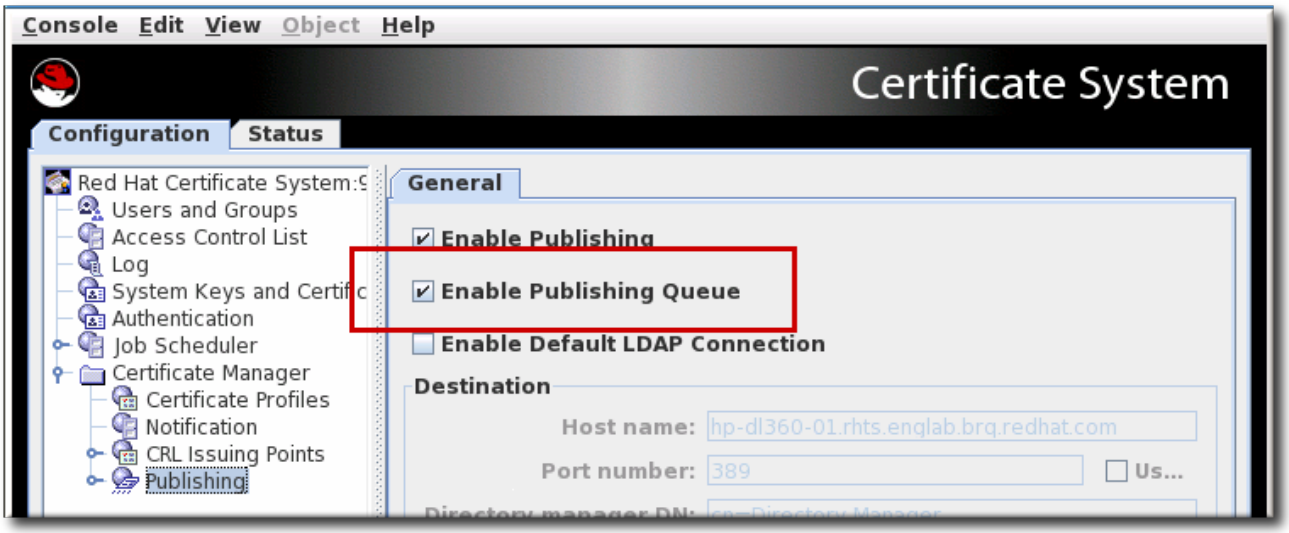
pkiconsole 已被弃用。



注意

虽然发布队列默认为禁用，但如果 控制台中 启用了 LDAP 发布，则队列会自动启用。否则，可以手动启用队列。

图 9.1. 启用 Publishing Queue



TIP

通过编辑 CS.cfg 文件启用发布队列，管理员可以设置用于发布操作的其他选项，如用于发布操作和队列页面大小的线程数量。

有关如何编辑 CS.cfg 文件配置此功能的说明，请参阅 Red Hat Certificate System 规划、安装和部署指南中的 启用和配置 发布 队列 部分。

9.8. 设置可恢复的 CRL 下载

证书系统提供中断 CRL 下载的选项，以便平稳恢复。这可以通过通过 HTTP 将 CRL 作为纯文本发布。此下载 CRL 的方法在检索 CRL 时具有灵活性，并降低整个网络拥塞。

9.8.1. 使用 wget 检索 CRL

由于 CRL 可以通过 HTTP 作为文本文件发布，所以可使用 wget 等工具手动从 CA 检索它们。wget 命令可用于检索任何公布的 CRL。例如，要检索比之前完整 CRL 更新的完整 CRL：

```
[root@server ~]# wget --no-check-certificate -d
https://server.example.com:8443/ca/ee/ca/crl/MasterCRL.bin
```

表 9.4 “用于检索 CRL 的 wget 选项” 中总结了 wget 的相关参数。

表 9.4. 用于检索 CRL 的 wget 选项

参数	描述
无参数	检索完整的 CRL。
-N	检索比本地副本更新的 CRL (delta 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。

9.9. 发布跨证书

跨对证书可以作为 LDAP 目录或文件的 crossCertificatePair 条目发布；这默认是启用的。如果禁用了此功能，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 Certificate Manager 链接，然后选择 Publishing 链接。

3. 单击 Publishing 下的 Rules 链接。这会在右侧打开 Rules Management 窗格。

4. 如果规则存在并已禁用，请选中 启用 复选框。如果删除了该规则，请单击 Add 并创建新规则。

   a. 从 类型 下拉菜单中选择 xcerts。

   b. 确保选择了 enable 复选框。

   c. 从 mapper 下拉菜单中选择 LdapCaCertMap。

   d. 从 publisher 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 都列在 CA 控制台左侧导航窗口中的 Publishing 链接下。在默认情况下，映射程序 LdapCaCertMap 指定了 crossCertificatePair 存储在 LdapCaSimpleMap LDAP 条目。默认情况下，发布者 LDAPCrossPairPublisher 将属性设置为 CA 条目中的跨密钥对证书到 crossCertificatePair;binary。

有关使用跨对证书的详情，请参考 第 17.5 节 “使用跨证书”。

有关创建跨对证书配置文件的更多信息，请参阅 Red Hat Certificate System 规划、安装和部署指南中的 配置跨Pair 配置集 部分。

注意

pkiconsole 已被弃用。

9.10. 测试发布到文件

验证证书管理器是否在发布证书，并且 CRL 正确发布到文件：

1. **打开 CA 的端到端页面，并请求证书。**

2. **如果需要，通过代理服务页面批准请求。**

3. **从终端实体页面检索证书，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 到 ASCII 工具，将 DER 编码的证书转换为其基本 64 编码格式。有关此工具的详情，请参考 BtoA (1) 手册页。**

   ```
   BtoA input_file output_file
   ```

   INPUT_ FILE 设置包含 DER 编码证书的文件的路径，output_file 设置要编写 base-64 编码证书的文件的路径。

6. **打开 ASCII 文件；base-64 编码证书与显示的证书类似：**

   ```
   -----BEGIN CERTIFICATE-----
   MMIIBtgYJYIZIAYb4QgIFoIIBpzCCAZ8wggGbMIIBRaADAgEAAgEBMA0GCSqGSIb3DQEBB
   AUAMFcxC
   AJBgNVBAYTAlVTMSwwKgYDVQQKEyNOZXRzY2FwZSBDb21tdW5pY2F0aWhfyyuougjgjjg
   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形式。有关此工具的更多信息，请参阅 PrettyPrintCert (1) 手册页。

```
PrettyPrintCert input_file [output_file]
```

INPUT_ FILE 设置包含 base-64 编码证书的 ASCII 文件的路径，以及 output_file （可选）设置要写入证书的文件的路径。如果没有设置输出文件，证书信息将写入标准输出。

8. 将输出与发布的证书进行比较；检查证书中的序列号与文件名中使用的序列号。

如果所有内容都匹配，则证书管理器配置正确，以将证书发布到文件。

9. 吊销证书。

10. 检查服务器是否生成了包含 CRL 的 DER 编码文件。

打开服务器要发布 CRL 作为二进制 blob 的目录。CRL 文件应具有名称，格式为 crl-this_update.der。this_update 指定来自 CRL 的时间 依赖此更新 变量的值。

11. 使用 Binary 到 ASCII 工具，将 DER 编码的 CRL 转换为其基本 64 编码格式。

```
BtoA input_file output_file
```

12. 使用 Pretty Print CRL 工具将基本 64 编码的 CRL 转换为可读形式。

```
PrettyPrintCrl input_file [output_file]
```

13. 比较输出。

9.11. 查看证书和 CRL 发布到文件

证书和 CRL 可以发布到两种类型的文件：base-64 编码或 DER 编码的。可以使用 dumpasn1 工具或 PrettyPrintCert 或 PrettyPrintCrl 工具将文件转换为用户友善格式来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用户友善格式。例如：

```
PrettyPrintCert example.bin example.cert
```

要查看 DER 编码文件的内容，只需使用 DER 编码的文件运行 dumpasn1、 PrettyPrintCert 或 PrettyPrintCrl 工具。例如：

```
PrettyPrintCrl example.der example.crl
```

9.12. 更新目录中的证书和 CRL

如果在 Directory 服务器停机时发布或撤销证书，则证书管理器和发布目录可能会变为不同步。当目录服务器备份时，需要手动发布或取消发布已发布或取消发布的证书。

要查找与目录不同步的证书 - 不在目录中的有效证书，且仍然在目录中撤销或过期的证书 - 证书管理器会在其内部数据库中保留证书记录是否已发布到目录中。如果证书管理器和发布目录不同步，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。

以下选择可用于将目录与内部数据库同步：

- 在内部数据库中搜索没有同步的证书，并发布或取消发布。
- 发布目录服务器停机时发布的证书。同样，未发布已撤销或在目录服务器停机时过期的证书。

- 根据序列号发布或取消发布一系列证书，从序列号 xx 到序列号 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。

9.12.1. 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 表单可用于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。

  通过调度自动化作业，可以从发布目录中删除过期的证书。详情请查看 第 13 章 设置自动化作业。
- 从目录中删除撤销的证书。

通过执行以下操作来手动更新目录：

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更新目录。如果更改非常大，则更新目录可能需要大量时间。在此期间，任何通过证书管理器所做的更改（包括签发或撤销的任何证书）都可能不包括在更新中。如果在更新目录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具有有效证书的目录时，可以使用为用户证书设置的发布规则来发布 CA 签名证书。这可返回对象类违反错误或其他映射器中的错误。选择适当的序列号范围来排除 CA 签名证书可以避免出现这个问题。CA 签名证书是 root CA 发布的第一个证书。

- 通过将 predicate 参数的值改为 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，将 predicate 参数设置为 profileId=caCACert，以发布从属 CA 证书。

9.12.2. 在目录中手动更新 CRL

证书管理器代理服务页面中的证书撤销列表 表单使用 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 所做的任何更改都不会包含在更新中。

更新目录后，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。

9.13. 注册自定义映射程序和过期插件模块

新的映射程序或发布程序插件模块可以在证书管理器发布框架中注册。可以删除不需要的映射程序或发布程序插件模块。在删除模块之前，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，自定义发布者插件名为 **MyPublisher.java**。

2. 编译新类。

```
javac -d . -classpath $CLASSPATH MyPublisher.java
```

3. 在 **CA** 的 **WEB-INF web** 目录中创建一个目录来保存自定义类，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的 *类* 目录中，并将所有者设置为证书系统用户**(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 **Publishing**。

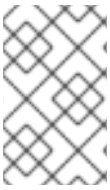
   c. 要注册映射程序模块，请选择 **Mappers**，然后选择 **映射程序插件注册** 选项卡。

   要注册发布者模块，请选择 **站s**，然后选择 **附件 插件注册** 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。类名称为实现 **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，若要在名为 **com.customplugins** 的软件包中注册名为

customMapper 的类，名称为 com.customplugins.customMapper。



注意

pkiconsole 已被弃用。

第 10 章 注册证书的身份验证

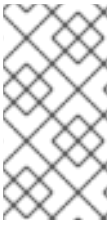
本章介绍了如何注册最终用户证书、创建和管理服务器证书、证书系统中可用的身份验证方法，以用于注册最终实体证书，以及如何设置这些身份验证方法。

注册 是将证书发布到结束实体的过程。这个过程正在创建和提交请求，验证请求它的用户，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。证书系统可以验证实体的三种方法：

- 在 代理批准的 注册中，最终用户请求将发送到代理以进行批准。代理批准证书请求。
- 在 自动注册 中，最终用户请求使用插件进行身份验证，然后处理证书请求；代理不涉及注册过程。
- 在 CMC 注册 中，第三方应用程序可以创建由代理签名的请求，然后自动处理。

证书管理器最初是为代理批准的注册和 CMC 身份验证配置。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

当通过配置自动通知为任何身份验证方法签发证书时，可以自动将电子邮件发送到结束实体。有关通知的详情，请查看 第 12 章 使用自动通知。

10.1. 配置代理应用注册

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，它发送到代理批准的代理队列。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，已签名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发送到代理批准的请求。

要使用代理批准的注册，请将身份验证方法留空到配置文件的 .cfg 文件中。例如：

```
auth.instance_id=
```

10.2. 自动注册

在自动注册中，当用户通过身份验证插件模块中设置的方法成功进行身份验证时，就会立即处理最终用户注册请求；不需要代理批准。提供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 和密码或其 DN 和密码对 LDAP 目录进行身份验证。请参阅 第 10.2.1 节 “设置基于目录的身份验证”。
- 基于 PIN 的注册。结束实体使用在其目录条目中设置的用户 ID、密码和 PIN 对 LDAP 目录进行身份验证。请参阅 第 10.2.2 节 “设置基于 PIN 的注册”。
- 基于证书的验证。某些类型的实体 - 最终用户和其他实体（如服务器或令牌）都使用 CA 发布的证书来证明其身份。这最常用于续订，其中显示了原始证书来验证续订过程。请参阅 第 10.2.3 节 “使用基于证书的身份验证”。
- AgentCertAuth.如果提交了请求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来作为代理进行身份验证。如果显示的证书被子系统识别为代理证书，则 CA 会自动处理证书请求。

  这种形式的自动身份验证可与证书配置文件关联，以注册服务器证书。

  此插件默认为启用，且没有参数。
- 基于平面文件的注册。专门用于路由器(SCEP)注册，使用文本文件，其中包含 IP 地址、主机名或其他标识符列表，通常是随机 PIN。路由器通过其 ID 和 PIN 验证 CA，然后将提供的凭据与文本文件的身份列表进行比较。请参阅 第 10.2.4 节 “配置平面文件身份验证”。

10.2.1. 设置基于目录的身份验证

UidPwdDirAuth 和 UdnPwdDirAuth 插件模块实施基于目录的身份验证。最终用户通过提供用户 ID 或 DN 和密码来为 LDAP 目录进行身份验证来注册证书。

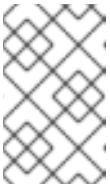
1. 创建 UidPwdDirAuth 或 UdnPwdDirAuth 身份验证插件模块的实例并配置实例。

   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。

   右侧窗格显示 Authentication Instance 选项卡，它列出了当前配置的身份验证实例。

   注意

   UidPwdDirAuth 插件默认为启用。

   c. 点击 Add。

   此时会出现 Select Authentication Plug-in Implementation 窗口。

   d. 选择 UidPwdDirAuth 用于用户 ID 和密码身份验证，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，或输入一个新名称。
   - dnpattern.指定代表主题名称模式的字符串，从目录属性和条目 DN 中公式。
   - ldapStringAttributes.指定应被视为最终实体 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令牌复制到身份验证令牌中，证书配置文件使

用它来生成主题名称。为此参数输入值是可选的。

- ldapByteAttributes.指定应该被视为最终实体的 LDAP 字节(binary)属性列表。如果指定，与这些属性对应的值将从身份验证令牌复制到身份验证令牌中，供其他模块使用，如向用户的证书添加其他信息。

  为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请求的 TCP/IP 端口；如果选择了 ldap.ldapconn.secureConn. 复选框，则这应该是 SSL 端口号。

- ldap.ldapconn.secureConn.指定身份验证目录侦听证书系统请求的端口的类型 SSL 或非 SSL。如果这是 SSL 端口，请选择。

- ldap.ldapconn.version.指定 LDAP 协议版本，可以是 2 或 3。默认值为 3，因为所有在版本 3.x 之后的 Directory 服务器都是 LDAPv3。

- ldap.basedn.指定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什么用户）和基本 DN 来构建 LDAP 搜索过滤器。

- ldap.minConns.指定允许到身份验证目录的最小连接数。允许的值是 1 到 3。

- ldap.maxConns.指定允许到身份验证目录的最大连接数。允许的值为 3 到 10。

f. 点确定。身份验证实例已设置并启用。

2. 通过为特定证书设置策略，将证书配置文件设置为用于注册用户。通过在证书配置文件中配置输入来自定义注册表单，并包含插件所需信息的输入来验证用户。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 “将 LDAP 目录属性值和其他信息插入到主题

Alt 名称”。

注意

pkiconsole 已被弃用。

设置 Bound LDAP 连接

有些环境需要禁止用于身份验证的 LDAP 服务器的匿名绑定。要在 CA 和 LDAP 服务器之间创建绑定连接，您需要进行以下更改：

- 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 cms.passwordlist 和 authPrefix 选项下使用的名称。

- 在 password.conf 中使用其密码从 CS.cfg 添加标签或提示：

```
externalLDAP=your_password
```

设置外部授权

也可以配置基于目录的身份验证插件，以评估用于身份验证的用户的组成员资格。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，它允许检索组。默认值为 false。
- 基于组的组 是组的基本 DN。当它与默认的 基于值不同时，需要指定它。
- groups 是组的 DN 组件。默认值为 ou=groups。

- groupObjectClass 是以下组对象类之一： groupofuniquenames,groupofnames.默认值为 groupofuniquenames。

- groupUseridName 是组对象成员属性中的用户 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，它决定是否为 userdn 或 ${groupUserIdName}=${uid} 属性搜索组对象成员属性。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，将配置集配置为使用 CS.cfg 中定义的 UserDirEnrollment auth 实例，以及根据组提供 ACL。例如：

```
auth.instance_id=UserDirEnrollment
auths.acl=group="cn=devlab-access,ou=engineering,dc=example,dc=com"
```

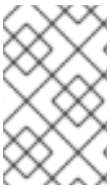
10.2.2. 设置基于 PIN 的注册

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，然后让用户在填充证书请求时提供 PIN 及其用户 ID 和密码。然后，用户会使用其用户 ID 和密码以及 LDAP 条目中的 PIN 根据 LDAP 目录进行身份验证。当用户成功验证时，请求会自动处理，并签发新证书。

证书系统提供了一个工具 setpin，它将 PIN 的必要模式添加到目录服务器，并为每个用户生成 PIN。

PIN 工具执行以下功能：

- 将 PIN 的必要模式添加到 LDAP 目录中。
- 为设置的 PIN 管理器用户添加具有读写权限的 PIN 管理器用户。
- 设置 ACI，以便在使用 PIN 后允许 PIN 移除，为 PIN 管理器授予 PIN 的读写权限，并防止用户创建或更改 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 所需的模式，将 PIN 添加到用户条目，然后将 PIN 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. 按照文件中介绍的说明进行适当的更改。

      通常，需要更新的参数是目录服务器的主机名、目录管理器的绑定密码和 PIN 管理器的密码。

   d. 使用指向 setpin.conf 文件的 optfile 选项运行 setpin 命令。

      ```
      setpin optfile=/usr/share/pki/native-tools/setpin.conf
      ```

      该工具使用新属性（默认为 pin）和新对象类（默认为 pinPerson）、创建一个 pinmanager 用户，并设置 ACI 来只允许 pinmanager 用户修改 pin 属性。

e.

要为特定用户条目或提供用户定义的 PIN 生成 PIN，请创建一个输入文件，其中包含列出这些条目的 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，或将值改为 no。

```
vim /usr/share/pki/native-tools/setpin.conf

setup=no
```

设置模式创建所需的 uers 和对象类，但工具不会在设置模式中生成 PIN。

g.

运行 setpin 命令，在目录中创建 PIN。

TIP

首先，测试工具没有 写入 选项来生成 PIN 列表，而无需实际更改目录。

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 命令会导致固定以纯文本形式存储在用户 LDAP 条目中。

h. 在完成设置所需的身份验证方法后，使用输出文件向用户发送 PIN。

确认基于 PIN 的注册正常工作后，向用户提供 PIN，以便在注册期间使用它们。要保护 PIN 的隐私，请使用安全、带外交付方法。

2. 在证书配置文件中为特定证书设置策略以注册用户。有关证书配置文件策略的详情，请查看 第 3 章 为颁发证书（证书配置文件）进行规则。

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。

右侧窗格显示 Authentication Instance 选项卡，它列出了当前配置的身份验证实例。

c. 点击 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 会限制用户一次注册多次，从而防止他们获取多个证书。
- pinAttr.指定 PIN 的身份验证目录属性。PIN Generator 实用程序将属性设置为 setpin.conf 文件中的 objectclass 参数的值；此参数的值是 pin。
- dnpattern.指定代表主题名称模式的字符串，从目录属性和条目 DN 中公式。
- ldapStringAttributes.指定应被视为最终实体 验证的 LDAP 字符串属性列表。为此参数输入值是可选的。
- ldapByteAttributes.指定应该被视为最终实体的 LDAP 字节(binary)属性列表。如果指定，与这些属性对应的值将从身份验证令牌复制到身份验证令牌中，供其他模块使用，如向用户的证书添加其他信息。

  为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证书系统请求的 TCP/IP 端口。
- 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。
- 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 3.x 之后的所有目录服务器版本都是 LDAPv3。
- ldap.ldapAuthentication.bindDN.指定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。建议单独用户条目，它只有权修改目录中的 PIN 属性。例如，请勿使用 Directory Manager 条目，因为它有修改整个目录内容的特权。

- *密码。提供与 ldap.ldapauthbindDN 参数指定的 DN 关联的密码。在保存更改时，服务器会将密码存储在单点登录密码缓存中，并使用它进行后续启动。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 SSL 客户端对要删除 PIN 的身份验证的证书 nickname。确保证书有效，并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 文件中已被正确配置，以正确将证书映射到目录中的 DN。这是仅删除 PIN 所必需的。*

- *ldap.ldapAuthentication.authtype.指定身份验证类型、基本身份验证或 SSL 客户端身份验证，以便从身份验证目录中删除 PIN。*

  - *Basic auth 指定基本身份验证。使用此选项时，为 ldap.ldapAuthentication.bindDN 和密码 参数输入正确的值；服务器使用来自 ldap. ldapAuthentication.bindDN 属性的 DN 来绑定到该目录。*

  - *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 客户端身份验证的证书 nickname。*

- *ldap.basedn.指定搜索身份验证目录的基本 DN；服务器使用来自 HTTP 输入的 uid 字段的值（用户在注册表单中输入的内容）和基本 DN 来构造 LDAP 搜索过滤器。*

- *ldap.minConns.指定允许到身份验证目录的最小连接数。允许的值是 1 到 3。*

- *ldap.maxConns.指定允许到身份验证目录的最大连接数。允许的值为 3 到 10。*

f. *点确定。*

4. *通过在证书配置文件中配置输入来自定义注册表单。包含插件验证用户所需的信息。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。*

注意

pkiconsole 已被弃用。

10.2.3. 使用基于证书的身份验证

基于证书的验证 是在显示证书来验证请求者的身份并自动验证和验证要提交的请求时。这最常用于续订进程，当原始证书由用户、服务器和应用程序提供，且该证书用于验证请求。

在有些情况下，使用基于证书的验证进行初始请求证书可能很有用。例如，令牌可以使用通用证书批量加载，然后用于在用户注册其用户证书时验证用户，或者用户可以被签发签名证书，然后用来验证其对加密密钥的请求。

基于证书的验证模块 SSLclientCertAuth 会被默认启用，此身份验证方法可在任何自定义证书配置文件中引用。

10.2.4. 配置平面文件身份验证

使用随机生成的 PIN 注册并验证路由器证书。CA 使用 flatFileAuth 身份验证模块来处理包含路由器身份验证凭据的文本文件。

10.2.4.1. 配置 flatFileAuth 模块

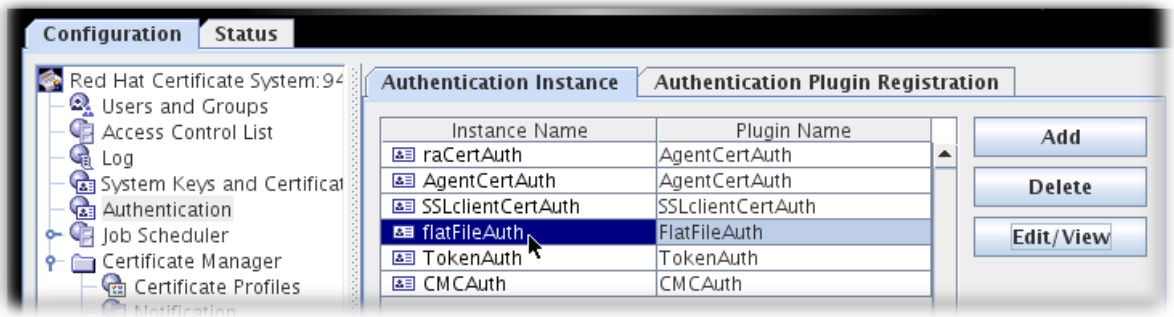
已为 SCEP 注册配置了平面文件身份验证，但可以编辑扁平文件的位置及其身份验证参数。

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。

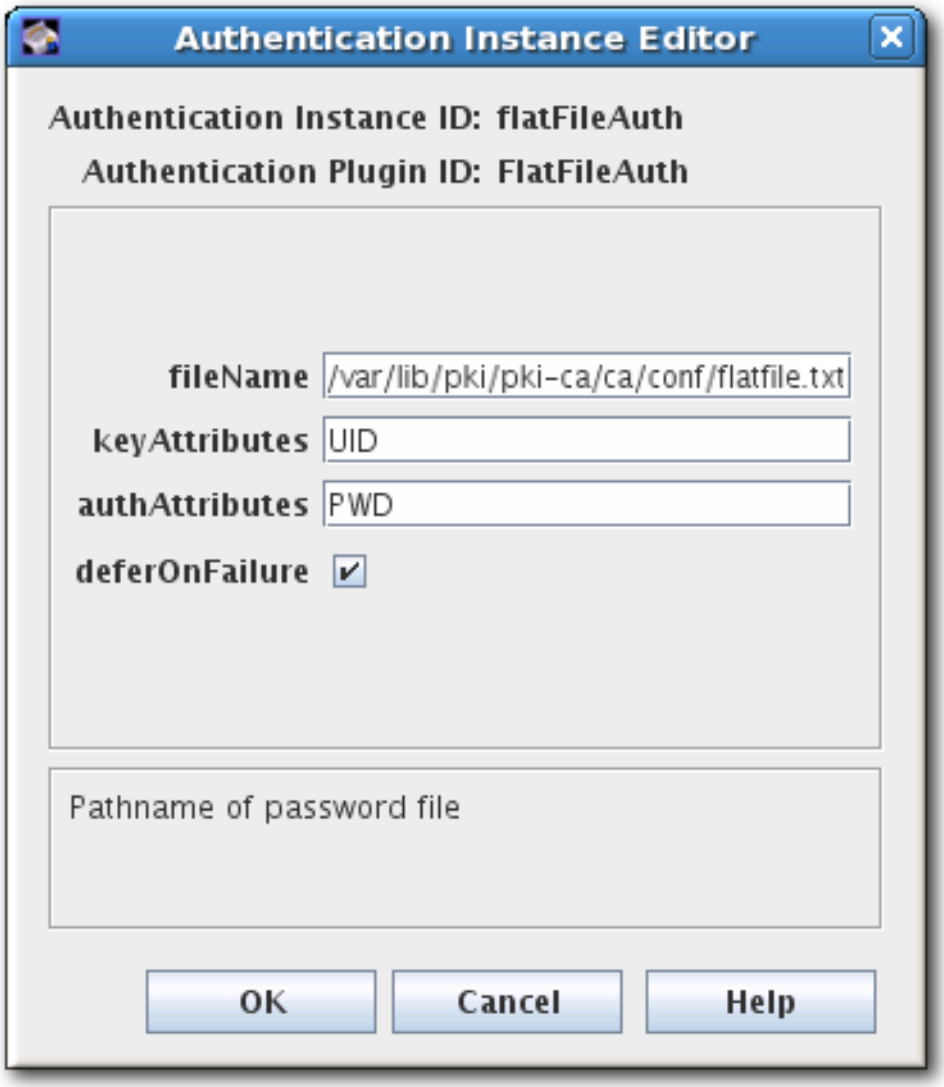
3. 选择 flatFileAuth 身份验证模块。



4. ***点 Edit/View。***

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 注册表单提交的另一个值，如 CN。也可以通过逗号（如 UID、CN）分隔多个 name 参数来使用它们。要更改 password 参数名称，请重置 authAttributes 字段。



6. 保存编辑。

注意

pkiconsole 已被弃用。

10.2.4.2. 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 PIN 时，都必须手动更新此文件。

默认情况下，此文件位于 /var/lib/pki/pki-ca/ca/conf/ 中，并为每个身份验证条目指定两个参数：站点 UID （通常是 IP 地址，可以是 IPv4 或 IPv6），以及路由器发布的 PIN。

```
UID:192.168.123.123
PIN:HU89dj
```

每个条目必须后跟一个空白行。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由空行分开，则当路由器尝试向 CA 进行身份验证时，它将会失败。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.
```

10.3. CMC 身份验证插件

CMC 注册允许注册客户端使用 CMC 身份验证插件进行身份验证，根据插件，证书请求是预签名的证书或用户证书。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 吊销。CMC 吊销允许客户端具有由代理证书签名的证书请求，或拥有证书的可验证用户，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 代理签署 CMC 请求时，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，其中包含与请求的证书相同的 subjectDN 属性。如果用户已经获得了一个签名证书，则只能使用用户签名的 CMC 请求，它可用于为其他证书证明用户身份。

基于 SharedSecret 的 CMC 请求意味着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 请求通常用于获取用户的

第一个签名证书，稍后用于获取其他证书。详情请查看 第 10.4 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

10.4. CMC SHAREDSECRET 身份验证

使用 Shared Secret 功能让用户向服务器发送未签名的 CMC 请求。例如，如果用户想要获取第一个签名证书，则需要此项。此签名证书稍后可用于为此用户的其他证书签名。

10.4.1. 创建共享 Secret 令牌

Red Hat Certificate System Planning, Installation, and Deployment Guide 中的 Shared Secret Workflow 部分描述了使用共享 Secret 令牌时的工作流。根据情况，最终用户或管理员会创建共享 Secret 令牌。

注意

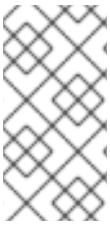
要使用共享机密令牌，证书系统必须使用 RSA 颁发保护证书。详情请参阅 RHCS 规划、安装和部署指南中的启用 CMC 共享 Secret 功能部分。

要创建共享 Secret Token，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，还要将 -h token_name 选项传给命令来设置 HSM 安全令牌名称。

有关 CMCSharedToken 工具的详情，请查看 CMCSharedToken(8) man page。



注意

生成的令牌是加密的，只有生成的用户知道密码。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。

创建共享令牌后，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 10.4.2 节 “设置 CMC 共享 Secret”。

10.4.2. 设置 CMC 共享 Secret

根据计划的操作，管理员必须在生成用户或证书的 LDAP 条目后存储共享 Secret 令牌。

有关工作流以及使用共享 Secret 的详细信息，请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide 中的 Shared Secret Workflow 部分。

10.4.2.1. 将 CMC 共享 Secret 添加到用于证书注册的用户条目中

要将 Shared Secret Token 用于证书注册，请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
```

10.4.2.2. 将 CMC 共享 Secret 添加到证书中以进行证书撤销

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
```

10.5. 测试注册

有关通过配置集测试注册的详情，请参考 第 3 章 为颁发证书（证书配置文件）进行规则。要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 end-entities 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. 输入了正确的密码时，客户端会生成密钥对。

   不要中断密钥生成过程。完成密钥生成后，请求将提交到服务器以发布证书。服务器会检查对证书配置文件的请求，仅在请求满足所有要求时发布证书。

   发布证书后，在浏览器中打开证书。

6. 验证证书是否已安装到浏览器的证书数据库中。

7. 如果使用 PIN 删除配置了基于 PIN 的目录身份验证，请使用相同的 PIN 重新注册另一个证书。请求应被拒绝。

10.6. 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 身份验证插件教程。

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保存自定义类，以便 CA 能够访问它们以进行注册表单。

```
mkdir /usr/share/pki/ca/webapps/ca/WEB-INF/classes
```

4. 将新插件文件复制到新的 类 目录中，并将所有者设置为证书系统用户(**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，点导航树中的 Authentication。

b. 在右侧窗格中，点 Authentication Plug-in Registration 选项卡。

这个选项卡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. 通过填写两个字段来指定要注册哪个模块：

- 插件名称。模块的名称。
- 类名称。此模块的类的完整名称。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 的软件包中注册名为 customAuth 的类，类名称为 com.customplugins.customAuth。

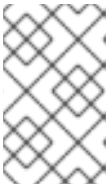
7. 注册该模块后，将模块添加为活跃的身份验证实例。

a. 在 Configuration 选项卡中，点导航树中的 Authentication。

b. 在右侧窗格中，点 Authentication Instance 选项卡。

c. 点击 Add。

d. 从列表中选择自定义模块 UidPwdDirAuthenticationTestms.java 来添加模块。填写该模块的适当配置。



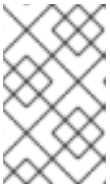
注意

pkiconsole 已被弃用。

8. 创建新的最终用户注册表单，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。



TIP

在编辑 CS.cfg 文件前备份它。

```
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。

```
pki-server restart instance_name
```

10.7. 使用命令行手动检查证书状态

要查看证书请求，请确保您作为代理进行身份验证，并具有适当权限来批准证书请求。有关配置 pki 命令行界面的详情，请参考 第 2.5.1.1 节 “pki CLI 初始化”。

查看请求：

1. 显示待处理证书请求列表：

```
$ 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它处于合法状态。然后，回答提示：如果请求有效，回答"批准，然后按 Enter 键。如果请求无效，回答 reject 并按 Enter。机构可以订阅 拒绝和取消 的语义差异；两者都不会发布任何证书。

10.8. 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
```

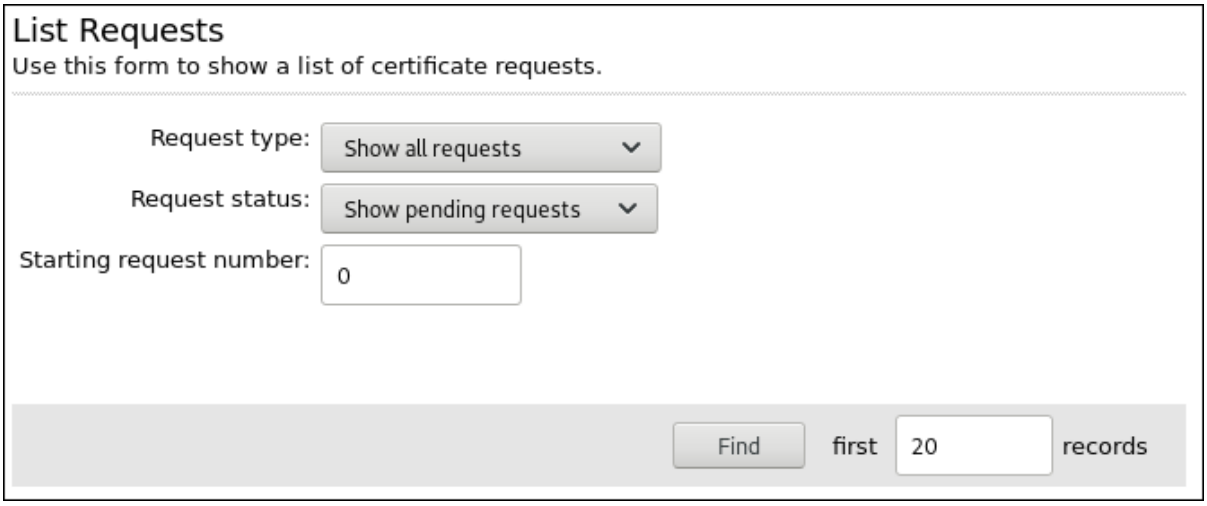
2. 作为代理进行身份验证。有关以用户身份验证并配置浏览器的详情，请参考 第 2.4.1 节 “浏览器初始化”。

3. 在左侧的侧边栏中，点 List requests 链接。

4.

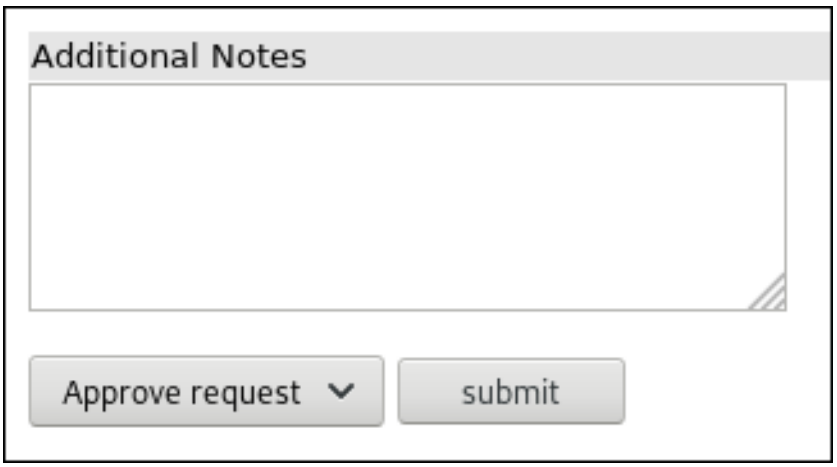
选择 **Show all requests for Request type 和 Show pending requests for Request status 来过滤请求。**

5. **点右下角的 Find。**



6. 结果页面列出了等待查看的所有待处理请求。点请求号来查看请求。

7. 检查请求信息并确保它是合法请求。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，如更改字段 无效。（可选）保留附加备注。



下拉菜单包含多个检查状态更新。选择 **Approve request 以批准请求或 拒绝请求 以拒绝该请求，然后单击 Submit。机构可以订阅拒绝请求的语义差异，取消请求 ；两者都不会发布任何证书。**

第 11 章 注册证书的授权(ACCESS EVALUATORS)

本章描述了使用访问等效器的授权机制。

11.1. 授权机制

除了身份验证机制外，还可以将每个注册配置集配置为具有自己的 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。访问评估是可插拔类，用于评估访问控制指令(ACI)条目。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、value），评估一个表达式，如 group='Certificate Manager Agents，并根据评估结果返回布尔值。

11.2. 默认评估

Red Hat Certificate System 提供四个默认的评估者。CS.cfg 文件中默认列出以下条目：

```
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，只允许 CA 代理使用该配置集注册：

```
authz.acl=group="Certificate Manager Agents"
```

ipaddress 访问 evaluator 评估请求主题的 IP 地址。例如，在以下注册配置集条目中，只有指定 IP 地址的主机才可以通过该配置集注册：

```
authz.acl=ipaddress="a.b.c.d.e.f"
```

用户访问 evaluator 评估用户 ID 以进行完全匹配。例如，在以下注册配置集条目中，只有与列出用户匹配的用户才可以使用该配置集进行注册：

```
authz.acl=user="bob"
```

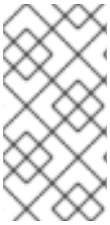
user_origreq 访问 evaluator 根据之前匹配的请求评估经过身份验证的用户是否相等。此特殊评估器专为续订目的而设计，以确保请求续订的用户是拥有原始请求的用户。例如，在以下续订注册配置集条目中，经过身份验证的用户的 UID 必须与请求续订的用户的 UID 匹配：

```
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 控制台注册。默认的 evaluator 可用作模板，以扩展到更目标插件中。

第 12 章 使用自动通知

证书系统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最终用户发送自动电子邮件通知。本章论述了自动通知，并详细介绍了如何启用、配置和自定义发送的通知电子邮件消息。



注意

由于可以发送的通知类型，只有证书管理器能够配置通知；此选项在其他子系统上不可用。

12.1. 关于 CA 的自动化通知

自动通知是在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确定何时发生特定事件；当事件发生时，系统会触发系统向配置的接收者发送电子邮件。每种通知都使用以纯文本或 HTML 的形式的模板来构建通知消息。模板包含扩展的文本和令牌，以填写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。HTML 模板也可以为不同的外观和格式进行自定义。

12.1.1. 自动通知的类型

有三种类型的自动通知：

- 证书发布.

  通知消息会自动发送到已签发证书的用户。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.

  当用户证书被撤销时，通知消息会自动发送到用户。

- Queue 中的请求。

  当请求使用为代理设置的电子邮件地址输入代理请求时，通知消息会自动发送到一个或多个代理。每次消息进入队列时，此通知类型都会发送电子邮件。有关队列作业中请求的更多信息，

请参阅 第 13.1.2.2 节 “requestInQueueNotifier (RequestInQueueJob)”。

还有一个作业，将通知发送到代理有关队列状态的代理，它包括队列状态的摘要（以秒为单位）。

12.1.2. 确定最终用户地址

通知系统通过检查第一个证书请求或撤销请求来确定最终实体的电子邮件地址，然后检查证书的主题名称，最后是证书的主题备用名称扩展（如果证书包含此扩展）。如果无法找到电子邮件地址，则会将通知发送到 Notification 面板的 Sender 的 Email Address 字段中指定的电子邮件地址。

12.2. 为 CA 设置自动通知

12.2.1. 在控制台中设置自动通知

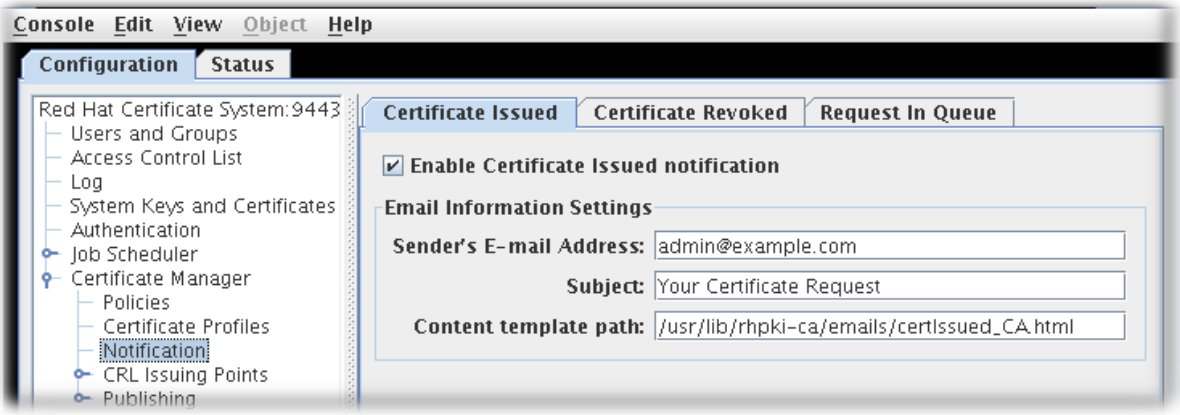
1. 打开 证书管理器控制台。

```
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 证书管理器 标题。然后选择 Notification。

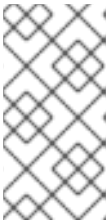
通知 选项卡会出现在窗口的右侧。



4. 对于三种类型的事件，可以发送通知：新发布的证书、撤销的证书和新证书请求。要为任何事件发送通知，请选择标签，选中 Enable 复选框，并在以下字段中指定信息：

- *发件人的电子邮件地址 .输入收到任何交付问题通知的用户的完整电子邮件地址。*
- *接收者的电子邮件地址 .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，请使用逗号分隔电子邮件地址。仅针对队列中的新请求。*
- *主题.输入通知的主题标题。*
- *内容模板路径.将路径（包括文件名）输入包含要构造消息内容的模板的目录。*

5. *点击 Save。*



注意

确保正确配置了邮件服务器。请参阅 第 12.4 节 “为证书系统通知配置邮件服务器”。

6. *自定义通知消息模板。请参阅 第 12.3 节 “自定义通知消息” 了解更多信息。*

7. *测试配置。请参阅 第 12.2.3 节 “测试配置”。*

注意

pkiconsole 已被弃用。

12.2.2. 通过编辑 CS.cfg 文件来配置特定通知

1. *停止 CA 子系统。*

```
pki-server stop instance_name
```

2. 为该实例打开 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。

   对于证书发送通知，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
   ```

   通知信息的参数在 第 12.2 节 “为 CA 设置自动通知” 中解释。

4. 保存该文件。

5. 重启 CA 实例。

   ```
   pki-server start instance_name
   ```

6. 如果创建了作业来发送自动消息，请检查邮件服务器是否已正确配置。请参阅 第 12.4 节 “为证书系统通知配置邮件服务器”。

7. 自动发送的消息可以自定义；如需更多信息，请参阅 第 12.3 节 “自定义通知消息”。

12.2.3. 测试配置

要测试子系统是否按照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配置中的电子邮件地址更改为可访问的代理或管理员电子邮件地址。

2. 打开端到端页面，并使用代理批准的注册表单请求证书。

   当请求排队代理批准时，应发送请求队列电子邮件通知。检查消息，以查看它是否包含配置的信息。

3. 登录代理接口，并批准请求。

   当服务器发布证书时，用户会收到证书发布的电子邮件通知到请求中列出的地址。检查消息以查看它是否有正确的信息。

4. 登录到代理接口并吊销证书。

   用户电子邮件帐户应包含读取证书已撤销的电子邮件消息。检查消息以查看它是否有正确的信息。

12.3. 自定义通知消息

电子邮件通知使用各种类型的消息的模板构建。这样，消息可以被告知、容易重复生成，且易于自定义。CA 使用模板进行通知消息。HTML 和纯文本消息都存在单独的模板。

12.3.1. 自定义 CA 通知消息

每种 CA 通知模板都有一个 HTML 模板和一个与其关联的纯文本模板。消息由文本、令牌和 HTML 模板构建，用于 HTML 标记。令牌是在构建消息时被当前值替换的消息中的数字符号($)标识的变量。有

关可用令牌列表，请参阅 表 12.3 “通知变量”。

可以通过更改消息模板中的文本和令牌来修改任何消息类型的内容。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。

certificate-issuance-notification 消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.
```

通过重新安排、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.
```

通知消息模板位于 /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，但证书被拒绝的模板除外；这些名称必须保持不变。与证书颁发和证书拒绝关联的模板必须位于同一目录中，且必须使用相同的扩展。

表 12.1 “通知模板” 列出为创建通知消息提供的默认模板文件。表 12.2 “作业通知电子邮件模板” 列出为创建作业摘要信息提供的默认模板文件。

表 12.1. 通知模板

filename	描述
certIssued_CA	在签发证书时，纯文本通知电子邮件的模板到结束实体。
certIssued_CA.html	在签发证书时，基于 HTML 的通知电子邮件的模板发送到结束实体。
certRequestRejected.html	当证书请求被拒绝时，基于 HTML 的通知电子邮件的模板到结束实体。
certRequestRevoked_CA	当证书被撤销时，用于纯文本通知电子邮件的模板到结束实体。
certRequestRevoked_CA.html	当证书被撤销时，基于 HTML 的通知电子邮件的模板到结束实体。
reqInQueue_CA	当请求进入队列时，用于纯文本通知电子邮件的模板。
reqInQueue_CA.html	当请求进入队列时，基于 HTML 的通知电子邮件的模板到代理。

表 12.2. 作业通知电子邮件模板

filename	描述
rnJob1.txt	用于公式发送到结束实体的消息内容的模板，以告知它们的证书即将过期，并在证书过期前续订或替换证书。
rnJob1Summary.txt	用于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包含的项目的模板，使用 **riq1Summary.html** 模板构建。
riq1Summary.html	用于公式报告或表的模板，它总结了证书管理器代理队列中待处理请求数。
publishCerts	总结了要发布到目录的证书的报告或表模板。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包含的项目的模板。

filename	描述
ExpiredUnpublishJob	总结了从目录中删除过期证书的报告或表模板。使用 **ExpiredUnpublishJobItem** 模板格式化表中的项目。
ExpiredUnpublishJobItem	用于格式化摘要表中包含的项目的模板。

表 12.3 “通知变量” 列出并定义通知消息模板中可以使用的变量。

表 12.3. 通知变量

令牌	描述
$CertType	指定证书类型；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(**other**)。
$ExecutionTime	提供作业运行的时间。
$HexSerialNumber	提供以十六进制格式发布的证书的序列号。
$HttpHost	提供最终实体应连接到的证书管理器的完全限定主机名，以检索其证书。
$HttpPort	提供证书管理器的端到端（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	提供有效期的结束日期。
$NotBefore	提供有效期的开始日期。
$RecipientEmail	提供接收者的电子邮件地址。
$RequestId	给出请求 ID。
$RequestorEmail	提供请求者的电子邮件地址。
$RequestType	提供发出的请求类型。

令牌	描述
$RevocationDate	提供证书撤销的日期。
$SenderEmail	提供发送者的电子邮件地址；这与通知配置中 **发送者电子邮件地址字段中指定** 的地址相同。
$SerialNumber	提供发布的证书的序列号；序列号在结果消息中显示为十六进制值。
$Status	提供请求状态。
$SubjectDN	提供证书主题的 DN。
$SummaryItemList	列出概述通知中的项目。每个项目对应于作业检测到续订或从发布目录中删除的证书。
$SummaryTotalFailure	提供概述报告中失败的项目总数。
$SummaryTotalNum	提供队列中待处理的证书请求总数，或者从概述报告的目录中更新或删除的证书的总数。
$SummaryTotalSuccess	显示摘要报告中项目总数的总数。

12.4. 为证书系统通知配置邮件服务器

通知和作业功能使用证书系统 CA 实例中配置的邮件服务器来发送通知消息。

在开始设置邮件服务器前，请确保在 CS.cfg 配置文件中指定以下参数：

```
smtp.host=localhost
smtp.port=25
```

通过执行以下操作来设置邮件服务器：

1. ***打开 CA 子系统管理控制台。例如：***

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

2. 在 Configuration 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域名，如 mail.example.com。默认情况下，邮件服务器的主机名是 localhost 而不是实际主机名。

   SMTP 邮件服务器侦听 25 的默认端口号。

4. 点击 Save。

注意

pkiconsole 已被弃用。

12.5. 为 CA 创建自定义通知

可以通过编辑证书系统 CA 的现有电子邮件通知插件来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前，请联系红帽支持服务。

第 13 章 设置自动化作业

证书系统提供可自定义的作业调度程序，它支持调度 cron 任务的各种机制。本章解释了如何配置证书系统，以使用特定的作业插件模块来完成作业。

13.1. 关于自动化作业

证书管理器控制台包含一个 Job Scheduler 选项，可在指定时间执行特定的作业。作业调度程序与传统的 Unix cron 守护进程类似，它会获取注册 cron 作业，并在预先配置的日期和时间中执行它们。如果配置，调度程序会按照指定间隔检查等待执行的作业；如果指定的执行时间已经到达，调度程序会自动启动作业。

作业作为 Java™ 类实施，然后作为插件模块注册到证书系统。作业模块的一个实现可用于配置作业的多个实例。每个实例必须具有唯一名称（没有空格的字母字符串），并且可以包含要应用到不同作业的不同的输入参数值。

13.1.1. 设置自动化作业

通过执行以下操作来设置自动化作业功能：

- 启用和配置作业调度程序；如需更多信息，请参阅 第 13.2 节 “设置作业调度程序”。
- 启用和配置作业模块并为这些作业模块设置首选项；如需更多信息，请参阅 第 13.3 节 “设置特定作业”。
- 通过更改与通知类型关联的模板来自定义与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。请参阅 第 12.3.1 节 “自定义 CA 通知消息” 了解更多信息。

13.1.2. 自动的作业类型

自动作业的类型是 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。

13.1.2.1. certRenewalNotifier (RenewalNotificationJob)

certRenewalNotifier 作业检查内部数据库中要过期的证书。找到证书的所有者时，它会自动发送电

子邮件，并继续在配置的时间段内发送电子邮件提醒，或直到证书被替换为止。该作业会收集所有续订通知的摘要，并将摘要发送给配置的代理或管理员。

作业决定使用电子邮件解析器发送通知的电子邮件地址。默认情况下，电子邮件地址位于证书本身或证书的相关注册请求中。

13.1.2.2. 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注册请求，作业会构建电子邮件消息总结其发现并将其发送到指定的代理。

13.1.2.3. publishCerts (PublishCertsJob)

publishCerts 作业检查已添加到尚未发布的发布目录中的新证书。添加这些新证书时，由 publishCerts 作业会自动发布到 LDAP 目录或文件。

注意

大多数时间，发布者会立即向适当的发布目录发布与规则匹配的任何证书。

如果在创建证书时成功发布证书，则 publishCerts 作业不会重新发布证书。因此，新证书不会列在作业摘要报告中，因为摘要仅列出了 publishCerts 作业发布的证书。

13.1.2.4. unpublishExpiredCerts (UnpublishExpiredJob)

过期的证书不会自动从发布目录中删除。如果证书管理器配置为将证书发布到 LDAP 目录，则目录将包含过期的证书。

未 发布的 Certs 作业检查已过期且仍然在配置的时间间隔内发布的证书。作业连接到发布目录并删除这些证书；然后在内部数据库中将 这些证书标记为未发布。作业收集已删除的过期证书的摘要，并将摘要发送给配置指定的代理或管理员。

注意

此作业会自动从目录中删除过期的证书。也可以手动删除过期的证书；有关此内容的更多信息，请参阅 第 9.12 节 “更新目录中的证书和 CRL”。

13.2. 设置作业调度程序

只有启用了作业调度程序时，证书管理器才能执行任务。作业设置（如启用作业调度、设置频率和启用作业模块）可以通过证书系统 CA 控制台或编辑 CS.cfg 文件来完成。

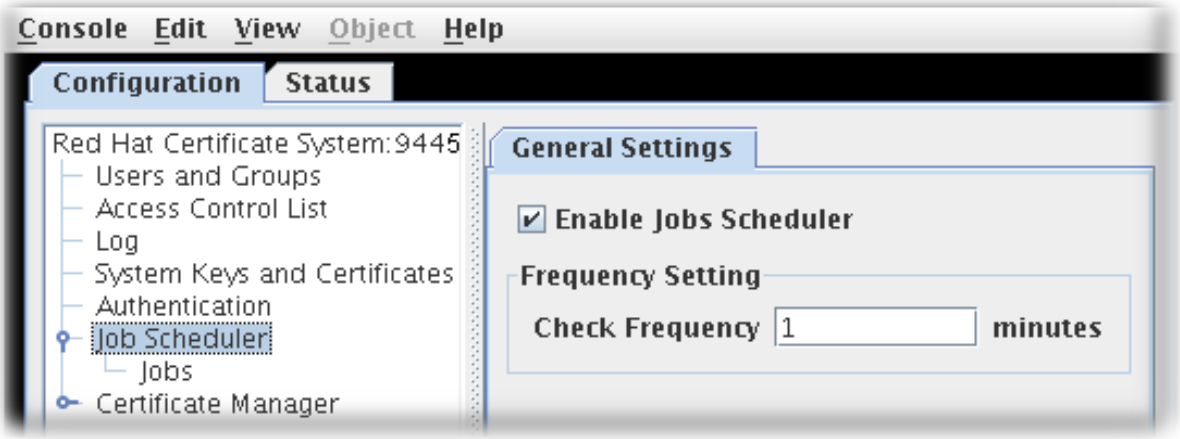
打开作业调度程序：

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。

这会打开 General Settings 选项卡，它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。

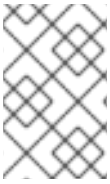
4. 在 Check Frequency 字段中设置调度程序检查作业的频率。

频率是作业调度程序守护进程线程唤醒的频率，并调用符合 cron 规格的配置作业。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。拖动证书管理器控制台的位置，以扩大整个窗口。

5. 点击 **Save**。



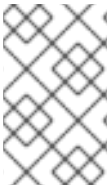
注意

pkiconsole 已被弃用。

13.3. 设置特定作业

自动化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。建议通过证书管理器控制台进行这些更改。

13.3.1. 使用证书管理器控制台配置特定作业



注意

pkiconsole 已被弃用。

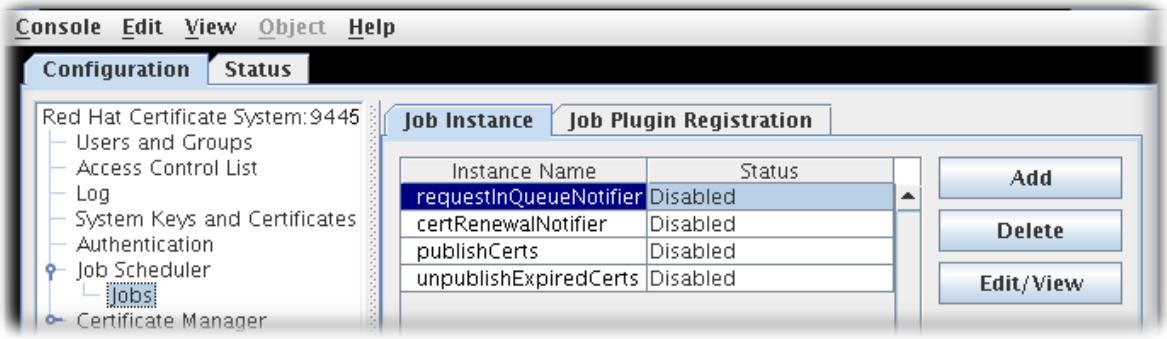
使用证书管理器控制台启用和配置自动化作业：

1. 打开 证书管理器控制台。

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

2. 确认启用了作业调度程序。请参阅 第 13.2 节 “设置作业调度程序” 了解更多信息。

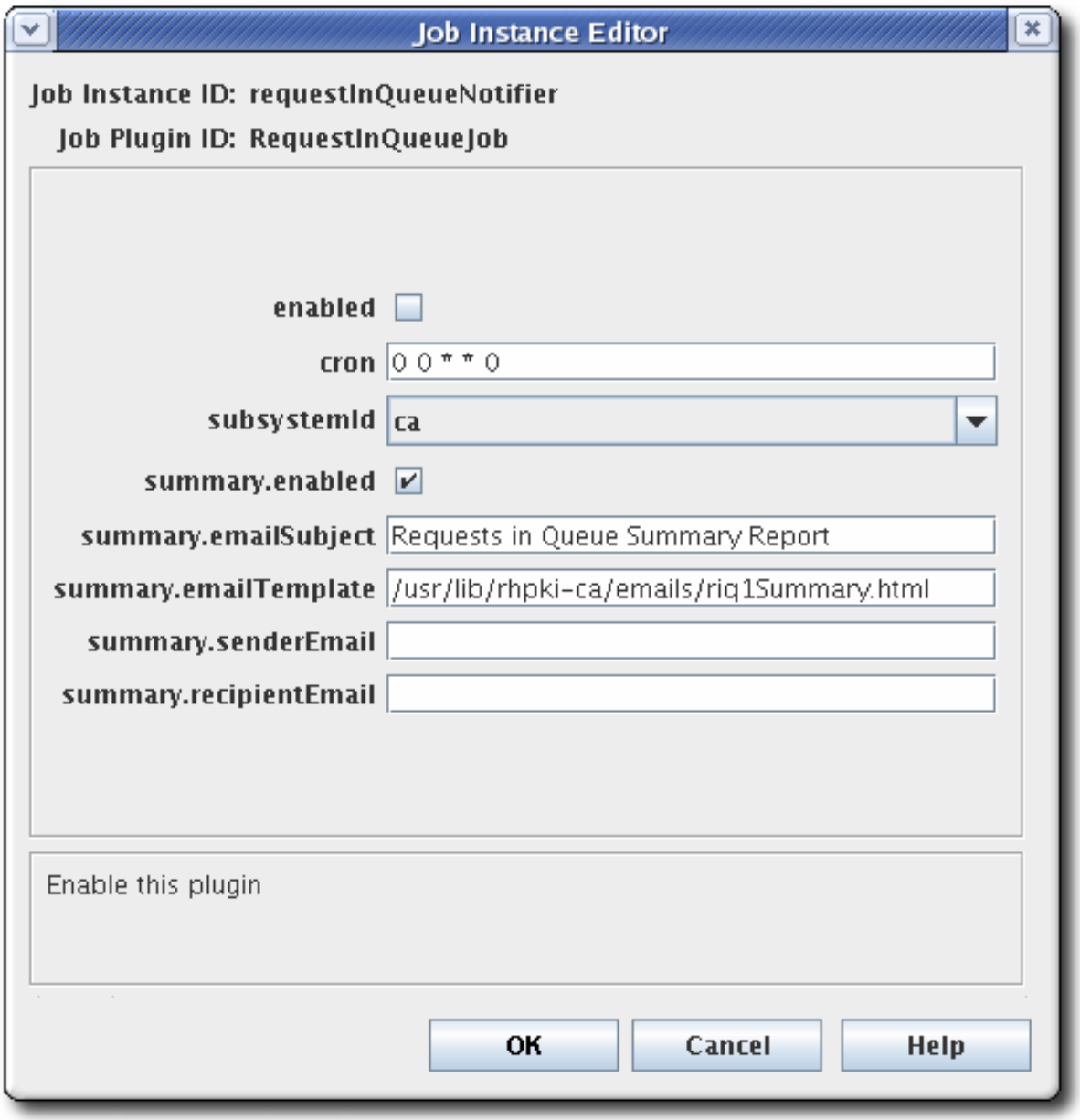
3. 在 **Configuration** 选项卡中，从导航树中选择 **Job Scheduler**。然后选择 **Jobs** 以打开 **Job Instance** 选项卡。



*从列表中选择作业实例，然后单击 **Edit/View**。*

*此时会打开 **Job Instance Editor**，显示当前的作业配置。*

*图 **13.1.** 作业配置*



4. 选择 enabled 以打开作业。

5. 通过在此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 13.3.3 节 “certRenewalNotifier 的配置参数”。
   - 有关 requestInQueueNotifier，请参阅 第 13.3.4 节 “requestInQueueNotifier 的配置参数”。
   - 有关 publishCerts，请参阅 第 13.3.5 节 “publishCerts 的配置参数”。
   - 有关 未发布的Certs，请参阅 第 13.3.6 节 “unpublishExpiredCerts 的配置参数”。
   - 有关设置 cron 时间频率的更多信息，请参阅 第 13.3.7 节 “自动任务的频率设置”。

6. 点确定。

7. 单击 Refresh 以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已正确设置。请参阅 第 12.4 节 “为证书系统通知配置邮件服务器”。

9. 自定义电子邮件文本和外观。

13.3.2. 通过编辑配置文件来配置作业

1. 确定启用了并配置作业调度程序；请参阅 第 13.2 节 “设置作业调度程序”。

2. 停止 CA 子系统实例。

```
pki-server stop instance_name
```

3. 在文本编辑器中打开该服务器实例的 CS.cfg 文件。

4. 编辑所配置作业模块的所有配置参数。

   - 要配置 certRenewalNotifier 作业，请编辑以 jobsScheduler.job.certRenewalNotifier 开头的所有参数，请参阅 第 13.3.3 节 “certRenewalNotifier 的配置参数”。
   - 要配置 requestInQueueNotifier 任务，请编辑以 jobsScheduler.job.requestInQueueNotifier 开头的所有参数，请参阅 第 13.3.4 节 “requestInQueueNotifier 的配置参数”。
   - 要配置 publishCerts 作业，请编辑以 jobsScheduler.job.publishCerts 开头的所有参数；请参阅 第 13.3.5 节 “publishCerts 的配置参数”。
   - 要配置 未发布的Certs 作业，请编辑以 jobsScheduler.job.unpublishExpiredCerts 开头的所有参数，请参阅 第 13.3.6 节 “unpublishExpiredCerts 的配置参数”。

5. 保存该文件。

6. 重启服务器实例。

```
pki-server start instance_name
```

7. 如果作业发送自动消息，请检查邮件服务器是否已正确设置。请参阅 第 12.4 节 “为证书系统通知配置邮件服务器”。

8. 自定义自动作业消息。

13.3.3. certRenewalNotifier 的配置参数

表 13.1 “certRenewalNotifier 参数” 提供可针对 certRenewalNotifier 作业配置的每个参数的详情，

可以在 CS.cfg 文件中或在证书管理器控制台中配置。

表 13.1. certRenewalNotifier 参数

参数	描述
enabled	指定作业是否启用或禁用。**true** 值启用作业；**false** 禁用它。
cron	设置应运行此任务时的计划。这会设置作业调度程序守护进程线程检查证书以发送续订通知的时间。这些设置必须遵循 第 13.3.7 节 "自动任务的频率设置" 中的惯例。例如： `0 3 * * 1-5` 示例中的作业在 3:00 pm 处运行周一到周五。
notifyTriggerOffset	在证书过期日期前设置发送第一次通知前的时长（以天为单位）。
notifyEndOffset	在证书没有替换时，设置在证书过期后，将继续发送通知的时长（以天为单位）。
senderEmail	设置通知消息的发送者，以通知任何发送问题。
emailSubject	设置通知消息的主题行的文本。
emailTemplate	将路径（包括文件名）设置为包含用于创建消息内容的模板的目录。
summary.enabled	设置是否应编译和发送续订通知的摘要报告。**true** 值启用发送概述；**false** 禁用它。如果启用，请设置剩余的摘要参数；服务器需要这些参数来发送摘要报告。
summary.recipientEmail	指定摘要消息的接收者。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置多个接收者。
summary.senderEmail	指定摘要消息的发送者的电子邮件地址。
summary.emailSubject	提供摘要消息的 subject 行。
summary.itemTemplate	将路径（包括文件名）提供给包含模板的目录，用于创建要为摘要报告收集的每个项目的内容和格式。
summary.emailTemplate	将路径（包括文件名）提供给包含用于创建摘要报告电子邮件通知的模板的目录。

13.3.4. requestInQueueNotifier 的配置参数

表 13.2 “requestInQueueNotifier 参数” 提供可针对 requestInQueueNotifier 作业配置的每个参数的详情，可在 CS.cfg 文件中或在证书管理器控制台中配置。

表 13.2. requestInQueueNotifier 参数

参数	描述
enabled	设置作业是否已启用(**true**)还是 disabled (**false**)。
cron	设置作业应运行时的时间调度。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。此设置必须遵循 第 13.3.7 节 “自动任务的频率设置” 中的惯例。例如： `0 0 * * 0`
subsystemid	指定运行作业的子系统。证书管理器唯一可能的值是 **ca**。
summary.enabled	指定是否应编译和发送作业摘要。**true** 值启用概述报告； **false** 禁用它们。如果启用，请设置剩余的摘要参数；服务器需要这些参数来发送摘要报告。
summary.emailSubject	设置摘要消息的 subject 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，其会收到任何发送问题的通知。
summary.recipientEmail	指定摘要消息的接收者。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，可以列出多个接收者。

13.3.5. publishCerts 的配置参数

表 13.3 “publishCerts Parameters” 提供每个 可发布Certs 作业配置的参数的详情，可在 CS.cfg 文件中或在证书管理器控制台中配置。

表 13.3. publishCerts Parameters

参数	描述
enabled	设置是否启用了作业。启用 **true** 值；禁用 **false**。

参数	描述
cron	设置作业运行时的时间调度。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证书的时间。此设置必须遵循 第 13.3.7 节 "自动任务的频率设置" 中的惯例。例如： `0 0 * * 6`
summary.enabled	指定作业发布的证书的摘要是否应编译和发送。**true** 值启用 summaries; **false** 可禁用它们。如果启用，请设置剩余的摘要参数；服务器需要这些参数来发送摘要报告。
summary.emailSubject	提供摘要消息的 subject 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，指向用于创建为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，该邮件将会收到任何交付问题的通知。
summary.recipientEmail	指定摘要消息的接收者。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，可以设置多个接收者。

13.3.6. unpublishExpiredCerts 的配置参数

表 13.4 "unpublishExpiredCerts Parameters" 在 CS.cfg 文件中或证书管理器控制台中为 未发布的 ExpiresCerts 作业配置各个参数的详情。

表 13.4. unpublishExpiredCerts Parameters

参数	描述
enabled	设置是否启用了作业。启用 **true** 值；禁用 **false**。
cron	设置作业运行时的时间调度。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证书的时间。此设置必须遵循 第 13.3.7 节 "自动任务的频率设置" 中的惯例。例如： `0 0 * * 6`

参数	描述
summary.enabled	指定作业发布的证书的摘要是否应编译和发送。**true** 值启用 summaries; **false** 可禁用它们。如果启用，请设置剩余的摘要参数；服务器需要这些参数来发送摘要报告。
summary.emailSubject	提供摘要消息的 subject 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，指向用于创建为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，该邮件将会收到任何交付问题的通知。
summary.recipientEmail	指定摘要消息的接收者。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，可以设置多个接收者。

13.3.7. 自动任务的频率设置

作业调度程序使用 Unix crontab 条目格式变体来指定检查作业队列和执行作业的日期和时间。如表 13.5 “调度作业的时间值” 和 图 13.1 “作业配置” 所示，时间条目格式由五个字段组成。（为 Unix crontab 指定的第六个字段不会被作业调度程序使用。） 值用空格或标签页分开。

每个字段可以包含单个整数或一对以连字符(-)分隔的整数，以指示包含的范围。要指定所有法律值，字段可以包含星号而不是整数。天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 13.5. 调度作业的时间值

字段	值
minute	0-59
hour	0-23
月份的日期	1-31
年月	1-12

字段	值
周天	0-6 （其中 0=Sunday）

例如，以下时间条目每小时指定 15 分钟(1:15、2:15、3:15 等)：

```
15 * * * *
```

以下示例设置在 4 月 12 日在 noon 上运行的作业：

```
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，以指定多个一天。如果指定了两天字段，则包含规格；也就是说，在一周的当日内不需要提供时间。例如，以下条目指定了每月第一和周五的执行时间：

```
0 0 1,15 * 1
```

要在没有另一个天的情况下指定一天类型，请在另一个日期字段中使用星号。例如，以下条目在每天的 3:15 点运行作业：

```
15 3 * * 1-5
```

13.4. 注册作业模块

自定义作业插件可以通过证书管理器控制台注册。注册新模块涉及指定模块名称和实施该模块的 Java™ 类的完整名称。

注册新作业模块：

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 目录中创建一个目录来保存自定义类，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的 类 目录中，并将所有者设置为证书系统用户(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. 登录证书管理器控制台。

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

   b. 在 Configuration 选项卡中，在左侧导航树中选择 Job Scheduler。选择 Jobs。

   此时会打开 Job Instance 选项卡，它列出了任何当前配置的作业。选择 Job Plugin Registration 选项卡。

   c. 点 Register 添加新模块。

   d. 在 Register Job Scheduler Plugin Implementation 窗口中，提供以下信息：

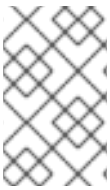
   - 插件名称。为插件模块输入一个名称。
   - 类名称。键入此模块的类全名；这是实施 Java™ 类的路径。如果这个类是软件包的一部分，请包含软件包名称。例如，若要注册名为 customJob 的类，它位于名为 com.customplugins 的软件包中，请输入 com.customplugins.customJob。

   e. 点确定。

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，请打开 作业插件注册 选项卡，如注册新模块时，选择要删除的模块，然后单击 Delete。出现提示时，确认删除。



注意

pkiconsole 已被弃用。

部分 IV. 管理子系统实例

第 14 章 基本子系统管理

本章讨论证书系统管理控制台、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、更改端口分配以及更改内部数据库。

14.1. PKI 实例

此版本的证书系统将继续为所有子系统支持 单独的 PKI 实例。

单独的 PKI 实例

- 作为基于 Java 的 Apache Tomcat 实例运行，
- 包含一个 PKI 子系统(CA、KRA、KRA、TKS 或 TPS)以及
- 如果位于同一物理虚拟机或虚拟机(VM)上在一起，则必须使用唯一端口。

此外，此版本的证书系统引入了 共享 PKI 实例 的概念。

共享 PKI 实例

- 作为基于 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
- 如果多个端口位于同一物理机器或虚拟机上，则必须使用唯一端口。

14.2. PKI 实例执行管理

启动、停止、重启或获取 PKI 实例的状态称为执行管理。每个 PKI 实例（单独的或共享）都已启动、停止、重启，其状态单独获得。本节介绍任何 PKI 实例的执行管理。

14.2.1. 启动、停止和重启 PKI 实例

使用 systemd 启动、停止和重新启动 PKI 实例。

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
```

3. 另外，您可以使用 **pki-server** 别名：

```
pki-server start|stop|restart instance_name
```

例如：

```
pki-server restart pki-tomcat
```

14.2.2. 在机器重启后重启 PKI 实例

如果意外关闭运行一个或多个 **PKI** 实例的计算机，则必须以正确顺序重启更多服务，以便子系统可以通过 **HTML** 服务页面和管理控制台使用。

1. 如果子系统使用的 **Directory** 服务器实例安装在本地计算机上，请重新启动管理服务器和目录服务器进程。

```
systemctl start dirsrv-admin.service
systemctl start dirsrv@instance_name.service
```

2. 启动证书系统子系统实例。

```
pki-server start instance_name
```

14.2.3. 检查 PKI 实例状态

systemctl 命令可用于检查进程的状态，显示它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，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
```

14.2.4. 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 Red Hat Administration Server、Directory 服务器和 CA：

```
# systemctl enable dirsrv-admin.service
# systemctl enable dirsrv.target
# systemctl enable pki-tomcatd@pki-tomcat.service
```

注意

使用 pkispawn 命令默认的 PKI 实例安装和配置会自动使实例在重启后启动。

要禁用此行为（即防止 PKI 实例在重启后自动启动），请运行以下命令：

```
# systemctl disable pki-tomcatd@pki-tomcat.service
# systemctl disable dirsrv.target
# systemctl disable dirsrv-admin.service
```

14.2.5. 为证书系统服务设置 sudo 权限

为了简化管理和安全性，可以配置证书系统和目录服务器进程，以便 PKI 管理员（而不是仅 root 用户）可以启动和停止服务。

设置子系统时推荐的选项是使用 pkiadmin 系统组。（请参阅 Red Hat Certificate System Planning, Installation and Deployment Guide。） 然后，所有作为证书系统管理员的操作系统用户都会被添加到此组中。如果这个 pkiadmin 系统组存在，则可以授予 sudo 访问权限来执行某些任务。

1. 编辑 /etc/sudoers 文件；在 Red Hat Enterprise Linux 8 中，可以使用 visudo 命令完成：

```
# visudo
```

2.

根据机器上安装的内容，为目录服务器、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

确保为机器上的每个证书系统、目录服务器和管理服务器设置 sudo 权限，并且仅对机器上的实例设置 sudo 权限。在一个计算机上可能有多个相同子系统类型的实例，或者没有子系统类型的实例。它取决于部署。

14.3. 打开子系统控制台和服务

每个子系统具有不同的接口来访问不同的用户类型。除 TKS 外，所有子系统都有某种类型的用于代理、管理员或最终用户（或全部三个）的网页。此外，CA、KRA、OCSP 和 TKS 都有一个基于 Java 的控制台，必须安装到服务器上，以执行管理任务来管理子系统本身。

可以自定义子系统的基于 Web 的服务页面的功能，以便更好地与机构的现有网站集成。请参阅 Red Hat Certificate System Planning、安装和部署指南。

14.3.1. 查找子系统网页

CA、KRA、OCSP、TKS 和 TPS 子系统具有用于代理、常规用户和管理员的网页。可以通过子系统的安全最终用户端口打开子系统主机的 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 服务页面都有可用服务页面列表；这些内容在 表 14.1 “默认网页” 中进行了概述。要专门访问任何服务，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（管理用户）Web 服务：

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

如果正确配置了 DNS，则可以使用 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它们，通常是与代理或管理员角色关联的接口。其他接口，即使那些在安全(SSL 连接)上运行的接口也不需要客户端身份验证。其中一些接口（如最终实体服务）可以配置为需要客户端身份验证，但其他接口无法配置为支持客户端身份验证。这些区别在 表 14.1 “默认网页” 中记录。

注意

任何人都可以访问子系统的最终用户页面，但访问代理或管理员网页都需要在 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。

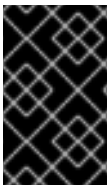
表 14.1. 默认网页

用于 SSL	用于客户端身份验证[a]	Web Services	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

用于 SSL	用于客户端身份验证[a]	Web Services	Web 服务位置
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
是	否	控制台（Console）	pkiconsole https://*host:port*/tks
令牌处理系统			
是		服务	index.cgi

[a] 可以重新配置具有客户端身份验证值 **No** 的服务，以要求客户端身份验证。没有 **Yes** 或 **No** 值的服务不能配置为使用客户端身份验证。

14.3.2. 启动证书系统管理控制台



重要

pkiconsole 已被弃用。

Console 通过 pkiconsole 命令通过其 SSL 端口连接到子系统实例来打开。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
```

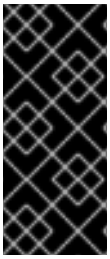
如果正确配置了 DNS，则可以使用 IPv4 或 IPv6 地址连接到控制台。例如：

```
pkiconsole https://1.2.3.4:8443/ca
pkiconsole https://[00:00:00:00:123:456:789:00:]:8443/ca
```

14.3.3. 为 Java 管理控制台启用 SSL

可以启用对证书系统控制台的基于证书的身份验证，以便管理员必须在登录到证书系统控制台之前使用客户端证书进行身份验证。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

如果为通过管理端口的客户端身份验证配置了 CA，并且 CA 是安全域管理器，则无法将 该 CA 用于其安全域的新 PKI 子系统。新的 PKI 实例通过管理端口将自身注册到安全域 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，将证书系统服务器设置为使用 SSL 客户端身份验证：

1. 使用此系统为任何管理员存储证书。证书应来自 CA 本身，或从哪个 CA 为子系统签名证书。

   a. 打开子系统控制台。

   b. 选择左侧的 Users and Groups 选项。

   c. 在 Users 选项卡中，选择管理用户，然后单击 Manage Certificates。

   d. 点 Import。

   e. 粘贴到 base-64 编码的 SSL 客户端证书，如存储在 Web 浏览器中的管理员证书。

确保客户端证书非常适合 SSL 客户端证书；否则，服务器将不接受客户端证书，并将在 /var/log/instanceID/system 中的错误日志中收到错误消息：

```
failure (14290): Error receiving connection
SEC_ERROR_INADEQUATE_CERT_TYPE - Certificate type not approved for application.)
```

2. 停止子系统。

```
pki-server stop instance_name
```

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 改为 sslclientauth ：

```
authType=sslclientauth
```

6. 保存该文件。

7. 打开 server.xml 文件。

8. 在 admin 接口连接器部分将 clientAuth="false" 属性改为 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 值表示客户端身份验证是首选的，但不需要。这允许客户端通过可轻松使用它的接口进行身份验证（如控制台），同时仍允许不支持客户端身份验证（安全域中的其他子系统）使用常规连接进行连接。

9. 启动子系统。

```
pki-server start instance_name
```

设置服务器后，将客户端配置为使用 SSL 客户端身份验证。

控制台必须有权访问用于向服务器 SSL 客户端身份验证的管理员证书和密钥。控制台的默认证书和密钥数据库存储在 .redhat-idm-console 目录中。

若要提供对管理员证书和密钥的访问权限，请将它们从管理员的浏览器导出到 .p12 文件，然后使用 pk12util 导入，或者将浏览器的证书和密钥数据库复制到 .redhat-idm-console 目录中。（这个过程假设证书是从浏览器导出到 .p12 文件。）

1. 将管理员用户证书和密钥从浏览器导出到文件中，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. 停止证书系统实例。

```
pki-server stop instance_name
```

5. 使用 pk12util 导入证书。

```
# pk12util -i /tmp/admin.p12 -d /user-directory/.redhat-idm-console -W [p12filepassword]
```

如果过程成功，命令会输出以下内容：

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

6. 从浏览器导出发布 CA 证书的 64 位 blob，并将它保存到 ca.crt 等文件中。

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. 启动证书系统实例。

```
pki-server start instance_name
```

9. 启动控制台；现在，它提示输入证书。

14.4. 在 JAVA 安全管理器下运行子系统

Java 服务可以选择使用安全管理器，为要执行的应用程序定义不安全和安全操作。安装子系统后，它们会自动启用 Security Manager，这意味着每个 Tomcat 实例都从运行的安全管理器开始。

14.4.1. 关于安全管理器策略文件

当在 Java 安全管理器内运行的五个 Java 子系统(CA、T OCSP、KRA、TKS 和 TPS)运行时，它们使用三组策略的组合：

- 位于 /usr/share/tomcat/conf 目录中的默认 Tomcat 策略中的 catalina.policy 文件；每当常规 Tomcat 文件更新时都会更新。
- pki.policy 文件，位于由 子系统实例提供的 /var/lib/pki/instance_name/subsystem_type/conf 目录中。
- 在 /var/lib/pki/instance_name/subsystem_type/conf 目录中的 custom.policy 文件，其中包含用户定义的安全策略。

当 Tomcat 服务开始创建修改后的 catalina.policy 文件时，这三个文件也会放在 /var/lib/pki/instance_name/subsystem_type/conf 目录中，它用于实例。

默认 pki.policy 文件包含授予 PKI 子系统使用的 Tomcat、LDAP 和 symkey 服务不受限制的访问权限。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；管理员可在该文件中写入策略，除了给定的 PKI 策略和 Tomcat 策略外，它还将使用该文件。

14.4.2. 启动没有 Java 安全管理器的子系统实例

PKI Tomcat 实例下配置的所有 Java 子系统均会在 Java 安全管理器下自动运行（除非通过覆盖 /etc/pki/default.cfg 文件中的 [Tomcat] 部分中的 pki_security_manager=true 创建实例）。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 实例，如下所示。

过程 14.1. 在不使用 Java 安全管理器的情况下启动实例

1. 停止该实例。

```
# pki-server stop instance_name
```

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。

```
# pki-server start instance_name
```

14.5. 配置 LDAP 数据库

证书系统执行证书和密钥管理功能以响应它收到的请求。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

为了实现这些功能，证书系统与红帽目录服务器合并，称为内部数据库或 本地数据库 。目录服务器作为证书系统配置的一部分引用；当配置证书系统子系统时，会在目录服务器中创建新的数据库。此数据库被证书系统实例完全用作嵌入式数据库，并可使用 Directory Server 附带的目录管理工具进行管理。

证书系统实例数据库与 serverRoot/slapd-DS_name/db/ 目录中的其他目录服务器数据库一起列出。这些数据库由 /etc/pki/default.cfg 文件(CS_instance_name-CA)中指定的子系统部分下的 pki_ds_database 变量的值来命名，默认为 CS_instance_name-CA、CS_instance_name-KRA、CS_instance_name-OCSP、CS_instance_name-TKS 和 CS_instance_name-TPS，这是实例配置中提供的默认格式。例如，对于名为 ca1 的证书管理器，数据库名称将是 ca1-CA。同样，数据库名称由 /etc/pki/default.cfg 文件的指定子系统部分下的 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-TKS 或 o=CS_instance_name-TPS 默认在配置中设置。

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 仅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系统数据。不要对其进行任何更改，或者将证书系统配置为使用任何其他 LDAP 目录。这样做可能会导致数据丢失。

此外，请勿将内部 LDAP 数据库用于任何其他目的。

14.5.1. 更改内部数据库配置

更改子系统实例用作其内部数据库的 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 Directory 服务器实例。

hostname 是安装目录服务器的机器的完全限定域名，如 **certificate. example.com**。证书系统使用此名称来访问目录。

默认情况下，用作内部数据库的目录服务器实例的主机名显示为 **localhost** 而不是实际主机名。这样做是为了使内部数据库在系统外可见，因为 **localhost** 上的服务器只能从本地计算机访问。因此，默认配置可最小化从本地机器外部连接到此目录服务器实例的风险。

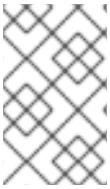
如果内部数据库的可见性限制为本地子网，可以将主机名更改为 **localhost** 以外的内容。例如，如果证书系统和目录服务器安装在用于负载平衡的独立机器上，请指定安装目录服务器的机器的主机名。

端口号是用于与目录服务器非 SSL 通信的 TCP/IP 端口。

DN 应该是目录管理器 DN。当证书系统子系统访问目录树与目录通信时，它会使用此 DN。

4. 点击 **Save**。

配置会被修改。如果更改需要重启服务器，则会显示包含该消息的提示。在这种情况下，重启服务器。



注意

pkiconsole 已被弃用。

14.5.2. 使用目录服务器中的证书系统发布的证书

要在安装证书系统时使用加密连接到目录服务器，需要使用由外部证书颁发机构(CA)或自签名证书发布的证书。但是，在设置证书系统 CA 后，管理员通常希望使用这个证书替换为证书系统发布的证书。

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，要生成使用 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，使 CA 能够处理请求：

   ```
   # 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. 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. ***通过 CMS (CMC)请求创建证书管理：***

i. ***创建包含以下内容的配置文件，如 ~/sslserver-cmc-request.cfg ：***

```
# 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.

*创建包含以下内容的配置文件，如 ~/**sslserver-cmc-submit.cfg** ：*

```
# 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. *显示目录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 服务器和 CA 证书的证书复制到目录服务器主机上。例如：*

```
# scp ~/ds.crt ~/certs_db/CA.pem ds.example.com:~/
```

g. *停止证书系统：*

```
# pki-server stop instance_name
```

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 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 安装服务器证书。

c. 启动 Directory 服务器实例：

```
# systemctl start dirsrv@instance_name
```

4. 启动证书系统：

```
# pki-server stop instance_name
```

5. （可选）配置基于证书的身份验证。详情请查看 第 14.5.3 节 “使用内部数据库启用 SSL/TLS 客户端身份验证”。

14.5.3. 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来对另一个实体进行身份验证。证书系统代理使用这种身份验证方法登录到代理服务页面，例如：

要在证书系统实例和它用作其内部数据库的 LDAP 目录实例之间使用 SSL/TLS 连接，必须启用客户端身份验证，以允许证书系统实例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。第一个是配置 LDAP 目录，如设置 SSL/TLS 并设置 ACI 来控制证书系统实例访问。第二个是在证书系统实例上创建用户，它将用于绑定到 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：安装带有安全 LDAP 连接的 PKI 子系统）。

14.5.4. 限制对内部数据库的访问

Red Hat Directory Server 控制台显示证书系统用作其内部数据库的 Directory 服务器实例的条目或图标。

与证书系统控制台不同，其访问仅限于具有证书系统管理员特权的用户，目录服务器控制台可以被任何用户访问。用户可以为内部数据库打开目录服务器控制台，并更改到其中存储的数据，例如从证书系统管理员组中删除用户或向组中添加自己的条目。

访问权限只能限制为知道目录管理器 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. 登录目录服务器控制台。
2. 选择 Certificate 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，然后选择 Pass-Through Authentication。
5. 在右侧窗格中，取消选择 Enable plugin 复选框。
6. 点击 Save。

   服务器提示重新启动服务器。
7. 点 Tasks 选项卡，然后点 重启 Directory Server。
8. 关闭 Directory 服务器控制台。
9. 服务器重启时，为内部数据库实例打开 Directory Server 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，内部数据库的目录服务器控制台才会打开。

14.6. 查看安全域配置

安全域 是 PKI 服务的 registry。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 registry 来查找其他服务。证书系统中的安全域服务同时管理证书系统子系统和一组共享信任策略的 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 位于同一安全域中，则可以安全地通信。

注意

安全域在子系统配置期间使用。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创建可信关系 - 类似于使用 TKS 和 KRA 的 TPS 和 KRA，则安全域用于在所选 TKS 和 KRA 实例上创建 TPS 代理用户。

registry 提供该域内子系统提供的所有 PKI 服务的完整视图。每个证书系统子系统必须是主机或安全域的成员。

只有 CA 可以托管和管理安全域。每个 CA 都有自己的 LDAP 条目，安全域是该 CA 条目下的机构组：

```
ou=Security Domain,dc=example,dc=com
```

然后，安全域组织组下每个子系统类型的列表，有一个特殊的对象类(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，并有一个特殊的 pkiSubsystem对象类来识别条目类

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
```

14.7. 管理用于子系统的 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，请参考为 Red Hat Enterprise Linux 8使用 SELinux 指南。

14.7.1. About SELinux

基本上，SELinux 识别系统中的 对象，可以是文件、目录、用户、进程、套接字或其他 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 都会映射到安全上下文，它定义了其对象类型以及如何在 Linux 服务器上正常工作。

系统进程在 SELinux 域中运行。每个域都有一组规则，用于定义 SELinux 域如何与系统上的其他 SELinux 对象交互。然后，这组规则决定了进程可以访问哪些资源以及如何对这些资源执行的操作。

对于证书系统，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属于同一个 SELinux 域，无论系统中的实例是什么，例如，如果服务器上安装了三个 CA，则所有三个 CA 都属于 http_port_t SELinux 域。

所有子系统的规则和定义都包含整个证书系统 SELinux 策略。安装子系统时，证书系统 SELinux 策略已配置，每次使用 pkispawn 添加子系统或使用 pkidestroy 被删除时，所有 SELinux 策略都会被更新。

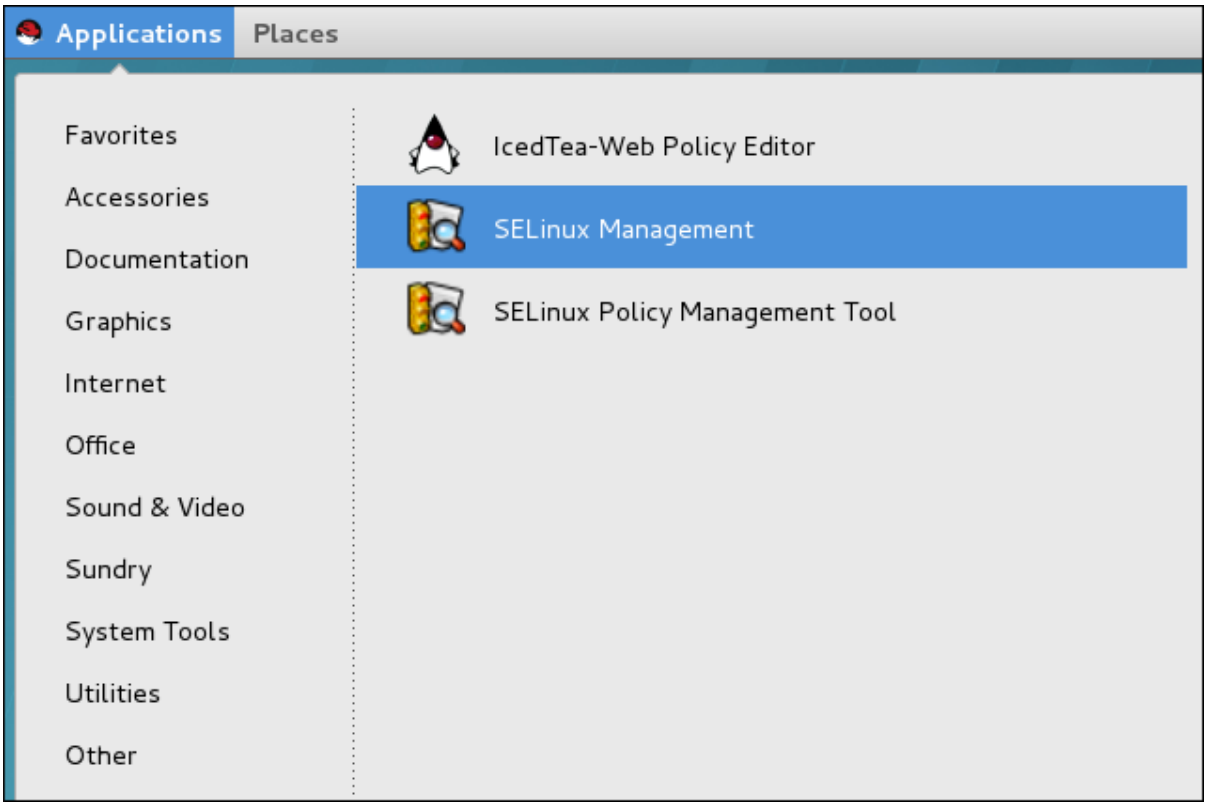
证书系统子系统以 enforcing 模式设置 SELinux，这意味着即使需要遵循所有 SELinux 规则，也可以成功执行证书系统操作。

默认情况下，证书系统子系统受 SELinux 策略限制运行。

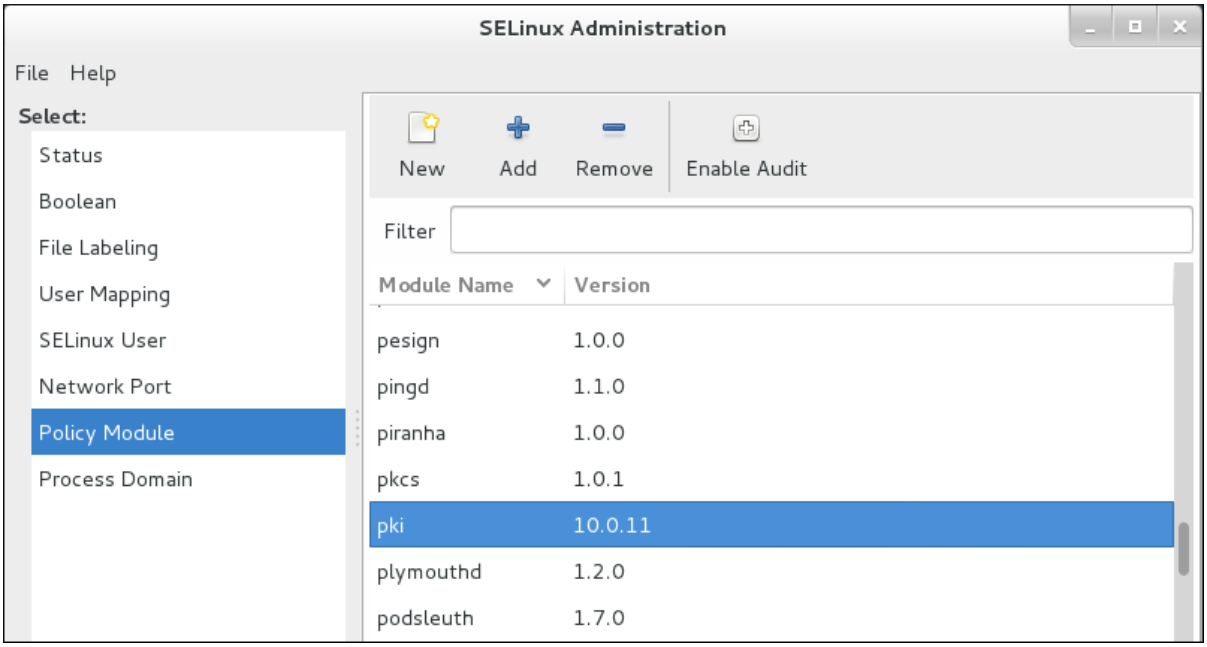
14.7.2. 查看 SELinux 策略以查看子系统

所有证书系统策略都是系统 SELinux 策略的一部分。可以使用 SELinux 管理 GUI 查看配置策略，您可以通过安装 policycoreutils-gui 软件包来获得这些策略。

1. 运行 system-config-selinux 命令或通过访问主系统菜单的 Applications → Other → SELinux Management 来打开工具。

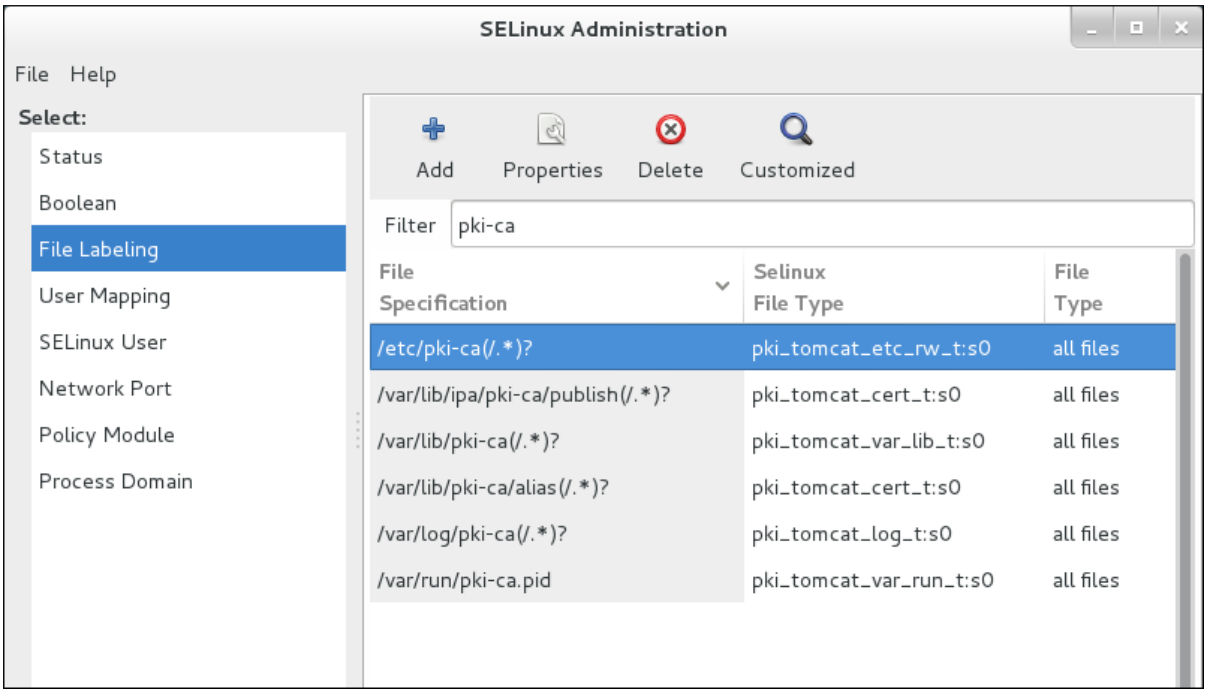


2. 要检查安装了证书系统 SELinux 策略的版本，请点左侧栏中的 Policy Module 部分。



3.

*要查看各个文件和进程上设置的策略，请点 **File Labeling** 部分。要查看子系统的端口分配的策略，请单击 **Network Port** 部分。*



14.7.3. 重新标记 nCipher netHSM 上下文

nCipher netHSM 软件不附带自己的 SELinux 策略，因此证书系统包含一个默认的 netHSM 策略，如 例 14.1 “netHSM SELinux Policy” 所示。

例 14.1. 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 thet 域与标记为 pki_common_t 和 pki_common_dev_t 的文件进行通信。

如果更改了任何 nCipher 配置（即使它位于默认目录中，/opt/nfast），请运行 restorecon 以确保所有文件都被正确标记：

```
restorecon -R /dev/nfast
restorecon -R /opt/nfast
```

如果在不同的位置安装 nCipher 软件，或者使用不同的 HSM，则需要使用 semanage 重新标记默认的证书系统 HSM 策略。

14.8. 备份和恢复证书系统

证书系统不包括备份和恢复工具。但是，证书系统组件仍然可以手动存档和恢复，如果证书或密钥信息丢失，则可能需要访问这些信息。在出现数据丢失或硬件故障时，需要定期备份证书系统的三个主要部分：

- 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流程。
- 安全数据库。安全数据库存储证书和密钥材料。如果它们存储在 HSM 上，请参阅 HSM 供应商文档来了解如何备份数据。如果信息存储在实例 别名 目录的默认目录中，则使用实例目录备份。要单独备份，请使用 tar 或 zip 等工具。
- 实例目录。实例目录包含所有配置文件、安全数据库和其他实例文件。这可以通过使用 tar 或 zip 等工具进行备份。

14.8.1. 备份和恢复 LDAP 内部数据库

Red Hat Directory Server 文档 包含有关备份和恢复数据库的更多详细信息。

14.8.1.1. 备份 LDAP 内部数据库

dsctl 命令的两对可用于备份目录服务器实例。每个 back-up 子命令都有一个对应部分来恢复它生成的文件：

- **db2ldif 子命令会创建一个 LDIF 文件，您可以使用 ldif2db 子命令恢复。**
- **db2bak 子命令会创建一个备份文件，您可以使用 bak2db 子命令恢复。**

14.8.1.1.1. 使用 db2ldif 备份

运行 **db2ldif 子命令可备份单个子系统数据库。**

注意

由于 db2ldif 子命令使用 dirsrv 用户运行，因此没有在 /root/ 目录下写入的权限，因此您需要提供可以写入的路径。

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，要备份主数据库，userRoot ：

1. **停止实例：**

```
# dsctl instance_name stop
```

2. **将数据库导出到 LDIF 文件中：**

```
# dsctl instance_name db2ldif userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
ldiffile: /tmp/example.ldif
[18/Jul/2018:10:46:03.353656777 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
[18/Jul/2018:10:46:03.383101305 +0200] - INFO - ldbm_back_ldbm2ldif - export userroot:
Processed 160 entries (100%).
[18/Jul/2018:10:46:03.391553963 +0200] - INFO - dblayer_pre_close - All database threads
now stopped
db2ldif successful
```

3.

3. 启动实例：

```
# dsctl instance_name start
```

要使用 ldif2db 子命令恢复 LDIF 文件，请参阅 第 14.8.1.2.1 节 “使用 ldif2db 恢复”。

14.8.1.1.2. 使用 db2bak 备份

运行 db2bak 子命令会备份那个目录服务器的所有证书系统子系统数据库（以及由该目录服务器实例维护的任何其他数据库）。例如：

例如：

1. 停止实例：

```
# dsctl instance_name stop
```

2. 备份数据库：

```
# dsctl instance_name db2bak
OK group dirsrv exists
OK user dirsrv exists
[18/Jul/2018:14:02:37.358958713 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
...
db2bak successful
```

3. 启动实例：

```
# dsctl instance_name start
```

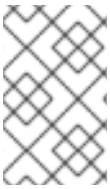
注意

由于 db2bak 子命令使用 dirsrv 用户运行，目标目录必须可由 dirsrv 写入。运行不带参数的子命令会在 /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 14.8.1.2.2 节 “使用 bak2db 恢复”。

14.8.1.2. 恢复 LDAP 内部数据库

根据您如何备份目录服务器实例，请使用 ldif2db 或 bak2db 和对应的文件来恢复数据库。



注意

在恢复数据库前，请确保停止实例。

14.8.1.2.1. 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 子命令导入文件。您可以指定一个数据库来从备份中恢复。例如，对于主数据库，userRoot ：

1. 停止 Directory 服务器实例：

```
# dsctl instance_name stop
```

2. 从 LDIF 文件中导入数据：

```
# dsctl instance_name ldif2db userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
[17/Jul/2018:13:42:42.015554231 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
...
[17/Jul/2018:13:42:44.302630629 +0200] - INFO - import_main_offline - import userroot:
Import complete.  Processed 160 entries in 2 seconds. (80.00 entries/sec)
ldif2db successful
```

3. 启动 Directory 服务器实例：

```
# dsctl instance_name start
```

14.8.1.2.2. 使用 bak2db 恢复

如果您创建带有 db2bak 的备份文件，请停止 Directory 服务器并使用 bak2db 子命令导入该文件。

例如：

1. **停止 Directory 服务器实例：**

```
# dsctl instance_name stop
```

2. **恢复数据库：**

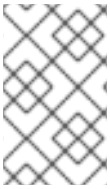
```
# dsctl instance_name bak2db /var/lib/dirsrv/slapd-instance_name/bak/instance_name-time_stamp/
OK group dirsrv exists
OK user dirsrv exists
[20/Jul/2018:15:52:24.932598675 +0200] - INFO - ldbm_instance_config_cachememsize_set - force a minimal value 512000
...
bak2db successful
```

3. **启动 Directory 服务器实例：**

```
# dsctl instance_name start
```

14.8.2. 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之前，停止子系统实例。

1. **停止子系统实例。**

```
pki-server stop instance_name
```

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. **重新启动子系统实例。**

```
pki-server start instance_name
```

如果数据损坏或硬件损坏，您可以使用证书系统备份文件(别名 数据库备份和完整实例目录备份)替换当前目录。要恢复数据，请使用 unzip 或 tar 工具解压缩存档文件，并将存档复制到现有文件中。

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。**

```
pki-server stop instance_name
```

3. 复制存档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. 确保恢复的文件的所有权和权限权限设置为 **pkiuser** ：

```
# chown -R pkiuser:pkiuser /var/lib/pki/pki-tomcat/ca/
```

5. 重新启动子系统实例。

```
pki-server start instance_name
```

14.9. 运行自测试

证书系统添加了允许自助范围服务器的功能。自我覆盖在启动时运行，也可以根据需要运行。当服务器启动，并在关键自测试失败时使服务器保持启动时运行启动自我tests。点子系统控制台中的自助范围按钮运行按需自助tests。

14.9.1. 运行自测试

CA、IADP、KRA 或 TKS 子系统的按需自测试从控制台运行。TPS 系统的按需自助范围从网页运行。

14.9.1.1. 从控制台运行自测试

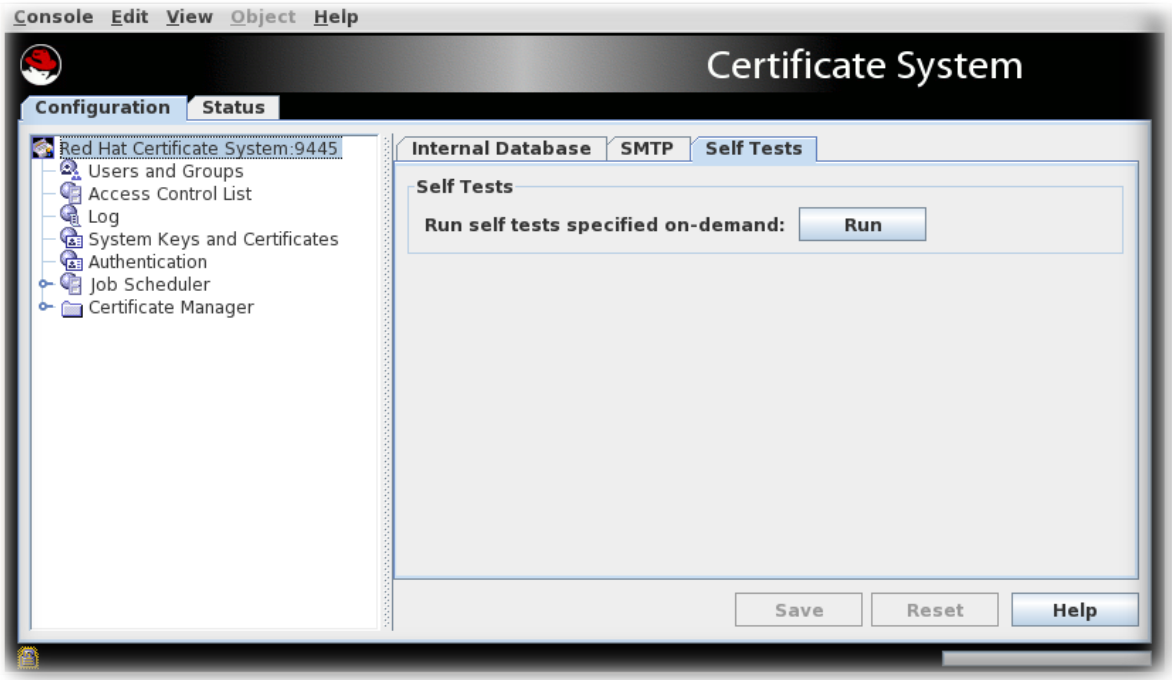
注意

pkiconsole 已被弃用。

1. 登录到控制台。

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

2. *选择左侧窗格中的子系统名称。*



3. *选择 **Self Tests** 选项卡。*

4. ***点 Run。***

   *为子系统配置的自助**tests** 将运行。如果有任何关键的自助限制失败，服务器将停止。*

5. *此时会出现 **On-Demand Self Tests Results** 窗口，显示此运行自**tests** 的日志记录事件。*

14.9.1.2. 运行 TPS Self-Tests

使用命令行界面(CLI)运行 TPS 自助范围：

- ***pki tps-selftest-find***
- ***pki tps-selftest-run***

- *pki tps-selftest-show*

14.9.2. 自我测试日志记录

一个单独的日志 selftest.log 被添加到日志目录中，其中包含启动自tests 和按需自助服务的报告报告。此日志是通过更改 CS.cfg 文件中的日志设置来配置。详情请参阅 Red Hat Certificate System 规划、安装和部署指南中的 修改自 测试配置部分。

14.9.3. 配置 POSIX 系统 ACL

POSIX 系统访问控制规则提供对系统用户权限的更精细的粒度。在完全配置后，必须为每个实例设置这些 ACL。有关 ACL 的详情，请查看 Red Hat Enterprise Linux 系统管理指南中的相应章节。

14.9.3.1. 为 CA、KRA、KRA、TKS、TKS 和 TPS 设置 POSIX 系统 ACL

ext4 和 XFS 等现代文件系统默认启用 ACL，在现代 Red Hat Enterprise Linux 安装中最有可能使用。

1. 停止该实例。

```
pki-server stop instance_name
```

2. 将组可读性设置为实例的目录和文件的 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/ 目录及其关联的文件中删除 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/ 目录及其关联的文件设置 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. 对 signedAudit/ 目录及其所有子目录重新应用执行(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。

```
pki-server start instance_name
```

8. 使用 getfacl 命令显示当前的 ACL 设置，确认正确应用了文件访问控制：

```
# 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 15 章 管理系统用户和组

本章解释了如何设置对管理、代理服务和端到端页面的访问权限的授权。

15.1. 关于授权

授权 是允许访问与证书系统关联的某些任务的过程。访问可以限制为允许特定用户或组对子系统的某些区域进行某些任务，以及不同的用户和组的不同任务。

用户特定于在其中创建它们的子系统。每个子系统都有自己的一组用户，独立于安装任何其他子系统。用户放置在组中，可以预定义或用户创建的组中。通过 访问控制列表 (ACL)将特权分配给组。管理控制台、代理服务界面和终端实体页面中的区域关联 ACL，这些页面在允许操作继续前执行授权检查。每个 ACL 中的 访问控制指令 (ACI)都会被创建，专门用于允许或拒绝该 ACL 到指定用户、组或 IP 地址可能的操作。

ACL 包含为创建的默认组的一组默认 ACI。可以修改这些 ACI，以更改预定义的组的权限，或为新创建的组分配特权。

授权通过以下流程：

1. 用户使用证书系统用户 ID 和密码或证书对接口进行身份验证。

2. 服务器通过将用户 ID 和密码与数据库中存储的用户或检查数据库中存储的证书匹配来验证用户。使用基于证书的身份验证时，服务器还通过将证书的 DN 与用户关联并检查用户条目来检查证书是否有效，并找到用户的组成员资格。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，然后通过将该用户 ID 与组中包含的用户 ID 关联来查找用户的组成员资格。

3. 当用户尝试执行操作时，授权机制比较用户的用户 ID、用户所属的组，或者用户的 IP 地址与为该用户、组或 IP 地址设置的 ACL 地址进行比较。如果存在允许该操作的 ACL，则操作将继续进行。

15.2. 默认组

用户的特权由用户的组（角色）成员资格决定。用户可以分配给的三个组（角色）：

- 管理员。这个组被授予对管理界面中所有可用任务的完整访问权限。

- 代理。这个组被授予对代理服务接口中所有可用的任务的完整访问权限。

- 审核员。这个组被授予查看签名的审计日志的访问权限。此组没有任何其他特权。

为子系统之间的通信而创建第四个角色。管理员不应为这样的角色分配实际用户：

- 企业管理员。在配置期间，当每个子系统实例加入到安全域时，会自动将其作为企业管理员分配特定于子系统的角色。这些角色自动提供安全域中子系统之间的可信关系，以便每个子系统可以有效地与其他子系统交互。

15.2.1. 管理员

管理员有权执行所有管理任务。用户通过添加到组的 Administrators 组来指定为管理员。该组的每个成员都具有该证书系统实例的管理特权。

必须为每个证书系统实例定义一个管理员，但实例可以拥有的管理员数量没有限制。配置了实例时会创建第一个管理员条目。

管理员使用其证书系统用户 ID 和密码通过简单的绑定进行身份验证。

表 15.1. 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。 • 管理域服务的访问控制。 默认情况下，托管域的 CA 的 CA 管理员被分配为安全域管理员。

角色	描述
企业 CA 管理员	• 自动批准域中任何 CA 的任何子 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	• 自动批准域中任何 CA 的任何 OCSP、服务器和子系统证书。 • 在安全域中注册和取消 OCSP 子系统信息。 • 将 CRL 发布信息推送到任何 CA。
Enterprise TKS 管理员	• 自动批准域中任何 CA 的任何服务器和客户端证书。 • 在安全域中注册和取消注册 TKS 子系统信息。
Enterprise TPS 管理员	• 自动批准域中任何 CA 的任何服务器和客户端证书。 • 在安全域中注册和取消注册 TPS 子系统信息。

如有必要，安全域管理员可以管理安全域和各个子系统上的访问控制。例如，安全域管理员可以限制访问，以便只有 KRA 部门才能设置 KRA 部门。

企业子系统管理员有足够的特权对域中的子系统执行操作。例如，企业 CA 管理员有权在配置过程中自动批准子 CA 证书。或者，安全域管理员也可以根据需要限制此右边。

15.2.2. 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。

一个审核员是通过将用户添加到 审核员 组，并将审核员的证书存储在用户条目中。审核员的证书用于加密用于为审计日志签名的密钥对的私钥。

在配置了子系统时，会设置 审核员 组。在配置过程中，不会将审核员分配给此组。

审核员使用其 UID 和密码通过简单的绑定向管理控制台进行身份验证。 经过身份验证后，审核员只能查看审计日志。它们无法编辑系统的其他部分。

15.2.3. 代理

代理是分配了最终用户证书和密钥权限的用户。代理可以访问代理服务接口。

代理通过将用户分配给适当的子系统代理组来创建，并且识别代理必须用于将 SSL 客户端身份验证用于子系统的证书，以便它能够服务来自代理的请求。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。
- 在线证书状态管理器代理组。
- Token Key Service Agents 组。
- 令牌处理系统代理组。

每个证书系统子系统都有自己的代理，其角色由子系统定义。每个子系统必须至少有一个代理，但子系统可以具有代理数量。

证书系统通过检查其内部数据库中用户的 SSL 客户端证书来识别并验证具有代理特权的用户。

15.2.4. 企业组

注意

不应该将实际用户分配给此组。

在子系统配置期间，每个子系统实例都会加入到安全域。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。这些角色自动提供安全域中子系统之间的可信关系，以便每个子系统可以有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 中，KRAs 来推送 KRA 连接器信息，CA 会自动批准 CA 中生成的证书。

企业子系统管理员有足够的特权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：

- 企业 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

此外，CA 实例有一个安全域管理员组，用于管理域中的安全域、访问控制、用户和信任关系。

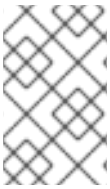
每个子系统管理员使用 SSL 客户端身份验证与安全域 CA 配置期间发布的子系统证书进行身份验证来对其他子系统进行身份验证。

15.3. 管理 CA、IADP、KRA 或 TKS 的用户和组

用户可以执行的操作由它们所属的组决定；例如，CA 的代理管理证书和配置文件，而管理员管理 CA 服务器配置。

四个子系统 - CA、IPP、KRA 和 TKS - 使用 Java 管理控制台来管理组和用户。TPS 具有基于 Web 的管理员服务，用户和组则通过其网页进行配置。

15.3.1. 管理组



注意

pkiconsole 已被弃用。

15.3.1.1. 创建新组

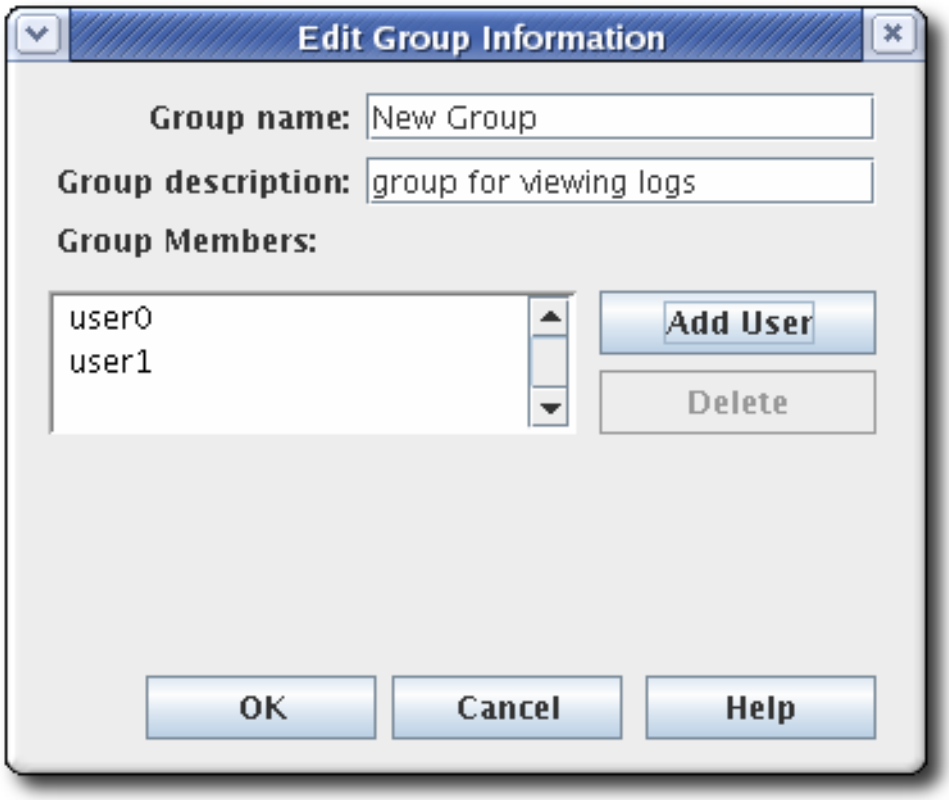
1. 登录到管理控制台。

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

2. 从左侧的导航菜单中选择 **Users and Groups**。

3. 选择 **Groups** 选项卡。

4. 单击 **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。请参阅 第 15.5.4 节 “编辑 ACL” 了解更多信息。如果没有将 ACl 添加到组的 ACL 中，则组不会对证书系统的任何部分没有访问权限。

15.3.1.2. 更改组中的成员

可以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 选项卡。
4. 从名称列表中选择组，然后单击 Edit。

5. 进行适当的更改。

- 要更改组描述，请在 Group description 字段中输入一个新的描述。
- 要从组中删除用户，请选择用户，然后单击 Delete。
- 要添加用户，请单击 Add User。从对话框中选择要添加的用户，然后单击确定。

15.3.2. 管理用户（管理员、代理和审核员）

每个子系统的用户单独维护。只是因为个人是一个子系统中的管理员，并不表示个人具有另一子系统的任何权限（甚至用户条目）。用户可以配置，并使用其用户证书，作为子系统的代理、管理员或审核员信任。

15.3.2.1. 创建用户

安装证书系统后，只有设置期间创建的用户已存在。这部分论述了如何创建用户。

注意

为安全起见，请为证书系统用户创建单个帐户。

15.3.2.1.1. 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，要将 example 用户添加到 CA 中：

```
# 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，要将 example 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
```

此命令将 CRMF 格式的证书签名请求(CSR)存储在 ~/user_name.req 文件中。

- 如果您的证书系统环境中没有密钥恢复授权(KRA)：

  - 创建 NSS 数据库目录：

```
# export pkiinstance=ca1
# echo ${pkiinstance}
# export agentdir=~/.dogtag/${pkiinstance}/agent1.dir
# echo ${agentdir}
# pki -d ${agentdir}/ -C ${somepwdfile} client-init
```

  - 将 CSR 存储在由 -o 选项指定的格式文件中，-d 用于初始化的 NSS 数据库目录的路径，-P 选项用于密码文件，-p 用于密码，-n 用于主题 DN：

```
# PKCS10Client -d ${agentdir}/ -P ${somepwdfile} -n "cn=agent1,uid=agent1" -o
${agentdir}/agent1.csr
PKCS10Client: Certificate request written into
/.dogtag/ca1/agent1.dir/agent1.csr
PKCS10Client: PKCS#10 request key id written into
/.dogtag/ca1/agent1.dir/agent1.csr.keyId
```

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 cert9.db, key4.db and pkcs11.txt
dbdir=~/.dogtag/pki-instance_name/

#password: password for cert9.db which stores the agent
#certificate
password=password

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

   根据您的环境以及上一步中使用的 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 (CMC)请求提交证书管理：

   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 cert9.db, key4.db and pkcs11.txt
#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 cert9.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.

向 CA 提交请求：

```
# 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）以用户身份将证书导入到自己的 ~/.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.

将证书添加到用户记录中：

a.

列出向用户发现证书的序列号的证书。例如，列出证书主题中包含 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
```

15.3.2.1.2. 使用控制台创建用户

注意

pkiconsole 已被弃用。

使用 PKI 控制台创建用户：

1.

登录到管理控制台。

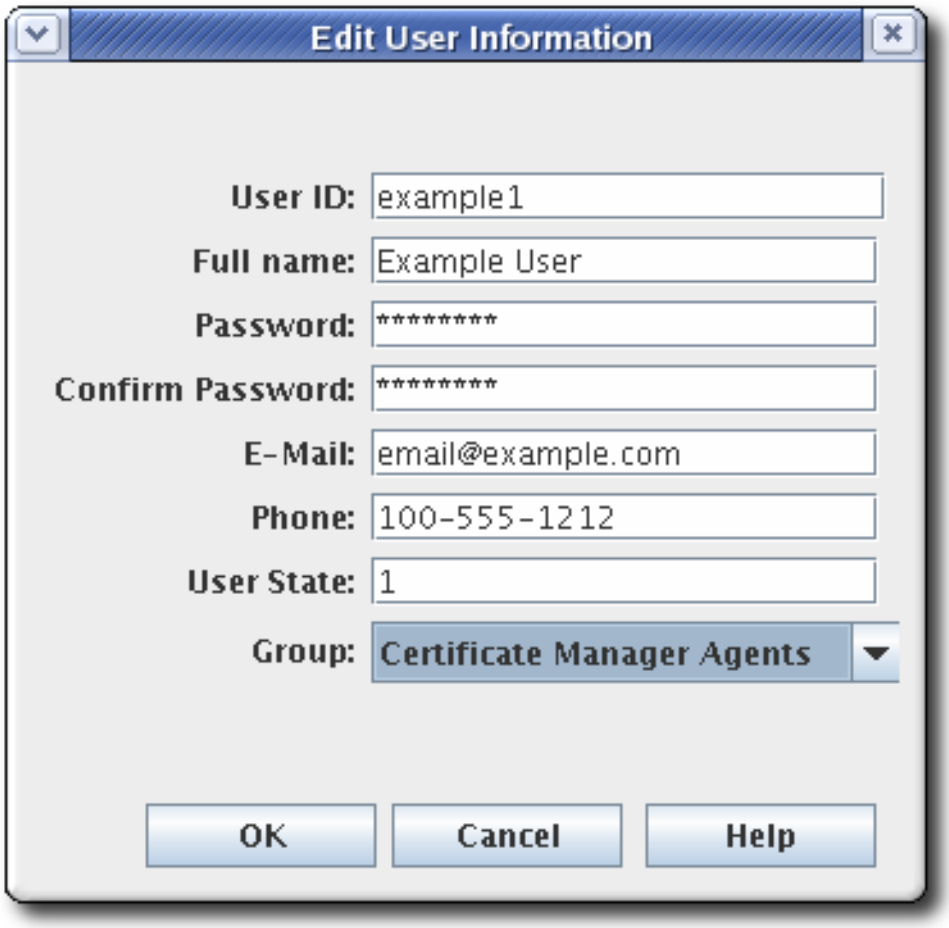
```
pkiconsole https://server.example.com:8443/subsystem_type
```

2.

在 Configuration 选项卡中，选择 Users and Groups。点击 Add。

3.

在 Edit User Information 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，用于添加用户的附加信息；最重要的是，此字段可以显示是否活动用户。

4.

选择用户所属的组。用户的组成员资格决定了用户拥有的权限。将代理、管理员和审核员分配给适当的子系统组。

5.

存储用户的证书。

a.

通过 CA 端到端服务页面请求用户证书。

b.

如果没有为用户配置文件配置自动注册，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 Certificates。

e.
单击 Import，并粘贴到 base-64 编码证书。

15.3.2.2. 更改证书系统用户的证书

1. 登录到管理控制台。
2. 选择 User and Groups。
3. 从用户 ID 列表中选择要编辑的用户，然后单击 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。包括 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE----- 标记行。

15.3.2.3. 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。重新生成证书会取其原始密钥及其原始配置集和请求，并使用新的有效期和过期日期重新创建相同的密钥。重新加密证书将初始证书请求重新提交到原始配置文件，但会生成一个新密钥对。管理员可以通过重新密钥来更新管理员证书。

每个子系统都有一个 bootstrap 用户，该用户在创建子系统时创建。在使用默认续订配置文件之一之前，可为此用户请求新证书。

可以使用原始证书的序列号直接在最终用户注册表单中续订管理用户的证书。

1. 以 CA 最终用户形式续订管理员用户证书，如 第 5.4.1.1.2 节 “基于证书的续订” 所述。这必须与首次发布的证书（或克隆）相同。

通过在结束日期页面中使用基于证书的续订表单，可以续订代理证书。自我续订用户 SSL 客户端证书。此表单可识别并更新存储在浏览器的证书存储中的证书。

注意

也可以使用 certutil 续订证书，如 第 17.3.3 节 “使用 certutil 续订证书” 所述。certutil 使用带有原始密钥的输入文件，而不是使用存储在浏览器中的证书启动续订。

2. 将更新的用户条目添加到内部 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 | 用户 | 管理员 | 证书 | 导入

   c. 在 Configuration 选项卡中，选择 Users and Groups。

   d. 在 Users 选项卡中，双击带有更新的证书的用户条目，然后单击 Certificates。

   e. 单击 Import，并粘贴到 base-64 编码证书。

注意

pkiconsole 已被弃用。

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中，方法是替换用户条目中的 userCertificate 属性，如 uid=admin,ou=body,dc=subsystem-base-DN。

15.3.2.4. 续订过期的管理员、代理和审核员用户证书

当有效的用户证书已过期时，您无法再使用 Web 服务页面或 pki 命令行工具需要身份验证。在这种情况下，您可以使用 pki-server cert-fix 命令来续订过期的证书。

在继续操作前，请确保：

- 您有一个有效的 CA 证书。
- 您有 root 权限。

过程 15.1. 续订过期的管理员、代理和审核员用户证书

1. 禁用自我测试。
   - 运行以下命令：

     ```
     # pki-server selftest-disable -i PKI_instance
     ```

   - 从 CA 的 CS.cfg 文件中删除以下行并重启 CA 子系统：

     ```
     selftests.container.order.startup=CAPresence:critical, SystemCertsVerification:critical
     ```

2. 检查客户端的 NSS 数据库中过期的证书，并查找证书的序列号(certificate ID)。
   a. 列出用户证书：

      ```
      # certutil -L -d /root/nssdb/
      ```

   b. 获取您要续订的过期证书序列号：

      ```
      # certutil -L -d /root/nssdb/ -n Expired_cert | grep Serial
          Serial Number: 16 (0x10)
      ```

3. 续订证书。本地 LDAP 服务器需要 LDAP 目录管理器的密码。

```
# pki-server cert-fix --ldap-url ldap://host389 --agent-uid caadmin -i PKI_instance -p
PKI_https_port --extra-cert 16
```

4. 重新提升的自我测试。

   - 运行以下命令：

     ```
     # pki-server selftest-enable -i PKI_instance
     ```

   - 或者将以下行添加到 CA 的 CS.cfg 文件中，并重启 CA 子系统：

     ```
     selftests.container.order.startup=CAPresence:critical, SystemCertsVerification:critical
     ```

要验证您在证书续订中是否成功，您可以运行以下命令来显示有关证书的足够信息：

```
# pki ca-cert-find
```

要查看特定证书的完整详情，包括属性、扩展、公钥 modulus、哈希等，您也可以运行：

```
# pki ca-cert-show 16 --pretty
```

15.3.2.5. 删除证书系统用户

用户可以从内部数据库中删除。从内部数据库中删除用户会从用户所属的所有组中删除该用户。要从特定组中删除用户，请修改组成员资格。

通过执行以下操作，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 Users and Groups。
3. 从用户 ID 列表中选择用户，然后单击 Delete。

4.
提示时确认删除。

15.4. 为 TPS 创建和管理用户

TPS 用户有三个定义的 角色，可作为 TPS 的组运行：

- 代理 执行实际令牌管理操作的代理，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，并对令牌有有限的控制
- 操作员，没有管理控制，但能够查看和列出通过 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（访问 TPS 的 Web 服务需要基于证书的验证），并且身份验证过程会检查 TPS 组条目 - ou=TUS Agents,ou=TUS Administrators, 和 ou=TUS Operators - 查看用户所属的角色，使用 Apache 的 mod_tokendb 模块。

TPS 的用户通过 Web UI 或 CLI 添加和管理。Web UI 可通过 https://server.example.com:8443/tps/ui/ 访问。

要使用 Web UI 或 CLI，TPS 管理员必须使用用户证书进行身份验证。

15.4.1. 列出和搜索用户

15.4.1.1. 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中。

3. 要搜索某些用户，请在搜索字段中写入关键字，然后按 Enter 键。若要再次列出所有用户，请删除关键字并按 Enter。

15.4.1.2. 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
```

要通过 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
```

15.4.2. 添加用户

15.4.2.1. 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 配置文件。

5. 点 Save 按钮。

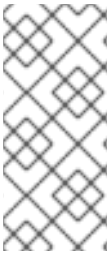
15.4.2.1.1. 从命令行

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
```

15.4.3. 为用户设置配置集

TPS 配置文件与 CA 配置文件非常相似；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性（如 CUID）自动分配给令牌。用户只能看到分配给它们的配置集的令牌。



注意

用户只能看到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。要使管理员能够搜索和管理 TPS 中配置的所有令牌，管理员用户条目应设置为 **All profile**。为用户设置特定的配置集是控制对特定用户或令牌类型的 Operator 和代理访问的简单方法。

令牌配置集是应用到令牌的策略和配置集。令牌配置文件根据令牌本身中的某种属性（如 CCUID 范围）自动映射到令牌。令牌配置文件作为 CA 配置集目录中的其他证书配置文件创建，然后添加到 TPS 配置文件 CS.cfg 中，以将 CA 的令牌配置文件映射到令牌类型。第 6.7 节 “映射解决程序配置” 中涵盖了配置令牌映射。

从 Web UI 管理用户配置文件：

1. 点 **Accounts** 选项卡。
2. 点 **Users** 菜单项。
3. 点击您要修改的用户的用户名。
4. 点 **Edit** 链接。
5. 在 **TPS Profile** 字段中，输入用逗号分开的配置集名称，或者输入 **All Profiles**。

6.
点 Save 按钮。

15.4.4. 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。组可以编辑，因此可以为用户添加和删除角色分配。

用户可以属于多个角色或组。例如， bootstrap 用户属于所有三个组。

15.4.4.1. 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.
单击您要更改的组名称，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。

15.4.4.2. 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
```

要通过 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
```

15.4.5. 管理用户证书

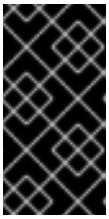
用户证书可以通过 CLI 管理：

- 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。第 5 章 请求、注册和管理证书 中解释了请求和提交证书。



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
```

15.4.6. 续订 TPS 代理和管理员证书

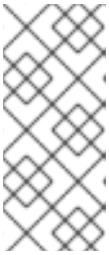
重新生成证书会取其原始密钥及其原始配置集和请求，并使用新的有效期和过期日期重新创建相同的密钥。

TPS 具有在创建子系统时创建的 bootstrap 用户。当使用其中一个默认续订配置文件时，可以为此用户请求新证书。

可以使用原始证书的序列号直接在最终用户注册表单中续订管理用户的证书。

1. 通过 CA 的最终用户表单续订用户证书，如 第 5.4.1.1.2 节 “基于证书的续订” 所述。这必须与首次发布的证书（或克隆）相同。

通过在结束日期页面中使用基于证书的续订表单，可以续订代理证书，即 自续订用户 SSL 客户端证书。此表单可识别并更新存储在浏览器的证书存储中的证书。



注意

也可以使用 certutil 续订证书，如 第 17.3.3 节 “使用 certutil 续订证书” 所述。certutil 使用带有原始密钥的输入文件，而不是使用存储在浏览器中的证书启动续订。

2. 向用户添加新证书并删除旧证书，如 第 15.4.5 节 “管理用户证书” 所述。

15.4.7. 删除用户

警告

可以删除 最后一个 用户帐户，操作无法撤消。对于选择要删除的用户，请非常小心。

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
```

15.5. 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。授权点在某些需要授权检查的操作组中定义。

15.5.1. 关于访问控制

访问控制列表 (ACL)是指定服务器操作的授权的机制。对于进行授权检查的每个操作，存在 ACL。可以在 ACL 中添加其他操作。

ACL 包含特定于允许或拒绝操作的 访问控制指令 (ACI)，如读取或修改。ACI 还包含 evaluator 表达式。ACL 的默认实现仅指定用户、组和 IP 地址，作为可能的等效类型。ACL 中的每个 ACI 指定是否允许或拒绝访问、允许或拒绝特定操作器，以及允许或拒绝哪些用户、组或 IP 地址才能执行操作。

通过更改与该用户所属的组关联的访问控制列表(ACL)或用户的 IP 地址来更改证书系统用户的权限。通过将该组添加到访问控制列表来分配新组。例如，只能查看日志的管理员的新组 LogAdmins 可以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。如果此组没有添加到任何其他 ACL 中，则此组的成员只能访问日志。

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 都显示在自己的行中。在这个接口窗口中，ACI 使用以下语法：

```
allow|deny (operation) user|group|IP="name"
```

注意

IP 地址可以是 IPv4 或 IPv6 地址。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。

例如，以下是允许管理员执行读取操作的 ACI：

```
allow (read) group="Administrators"
```

ACI 可以配置多个操作或操作。该操作以逗号分隔，在两端都没有空格。例如：

```
allow (read,modify) group="Administrators"
```

ACI 可以通过使用两个管道符号(||)分隔多个组、用户或 IP 地址，并在两端使用空格。例如：

```
allow (read) group="Administrators" || group="Auditors"
```

管理控制台可以创建或修改 ACI。接口设置是否在 Allow 和 Deny 字段中允许或拒绝操作，设置 Operations 字段中可以执行的操作，然后在 Syntax 字段中列出被授予或拒绝访问权限的组、用户或 IP 地址。

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。如果没有允许包含用户 ID、组或 IP 地址的 ACI，则组、用户 ID 或 IP 地址将被拒绝访问。

注意

如果用户没有明确允许访问某一资源的任何操作，则此用户被视为被拒绝；不需要拒绝访问。

例如，user JohnB 是 Administrators 组的成员。如果 ACL 只有以下 ACL，则 JohnB 将被拒绝任何访问，因为他与任何 allow ACI 不匹配：

```
Allow (read,modify) group="Auditors" || user="BrianC"
```

通常不需要包含 deny 语句。然而，在某些情况下可能会出现，当指定情况时很有用。例如： JohnB 是 Administrators 组的成员，刚刚触发。如果用户无法立即删除，则可能需要拒绝对 JohnB 的访问。另一个情况是，一个用户是 192.168.1.0/24 C，但不应该更改某些资源。由于 Administrators 组必须访问此资源，因此可以通过创建拒绝此用户访问权限的 ACI 来专门拒绝访问。

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该操作。为 ACL 设置的操作因 ACL 和子系统而异。可以定义的两个通用操作是读取和修改的。

ACI 编辑器的语法字段为表达式设置 evaluator。evaluator 可以指定组、名称和 IP 地址(IPv4 和 IPv6 地址)。它们与实体名称一起指定为等号(=)或不相等(!=)。

ACL 中包含组的语法是 group="groupname"。排除组的语法为 group!="groupname"，其允许除命名的组以外的任何组。例如：

```
group="Administrators" || group!="Auditors"
```

也可以使用正则表达式来指定组，例如使用通配符字符，如使用星号(DSL)。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。

ACL 中包括用户的语法是 user="userID"。排除用户的语法为 user!="userID"，它允许除命名的用户 ID 之外的任何用户 ID。例如：

```
user="BobC" || user!="JaneK"
```

要指定所有用户，请提供 任何body 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，如使用星号(DSL)。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。

ACL 中包含 IP 地址的语法是 ipaddress="ipaddress"。从 ACL 中排除 ID 地址的语法是 ipaddress!="ipaddress"。IP 地址通过其数字值指定；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。IPv4 地址的格式为 n.n. n.n 或 n.n. n.n,m.m.m，子网掩码为 n. n.n.n。IPv6 地址使用 128 位命名空间，其 IPv6 地址用冒号和以句点分开的子网掩码分开。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，如使用通配符字符，如使用星号(DSL)。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。

通过使用两个管道字符(||)分隔每个值，可以在两端使用空格来创建带有多个值的字符串。例如：

```
user="BobC" || group="Auditors" || group="Administrators"
```

15.5.2. 更改子系统的访问控制设置

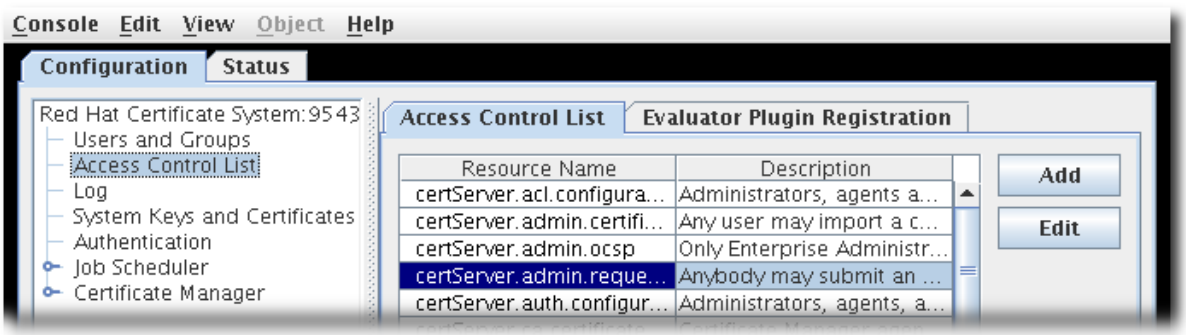
有关如何通过编辑 CS.cfg 文件配置此功能的说明，请参阅 Red Hat Certificate System 规划、安装和部署指南中的 更改子系统的访问控制设置 部分。

15.5.3. 添加 ACL

ACL 存储在内部数据库中，且只能在管理控制台中进行修改。

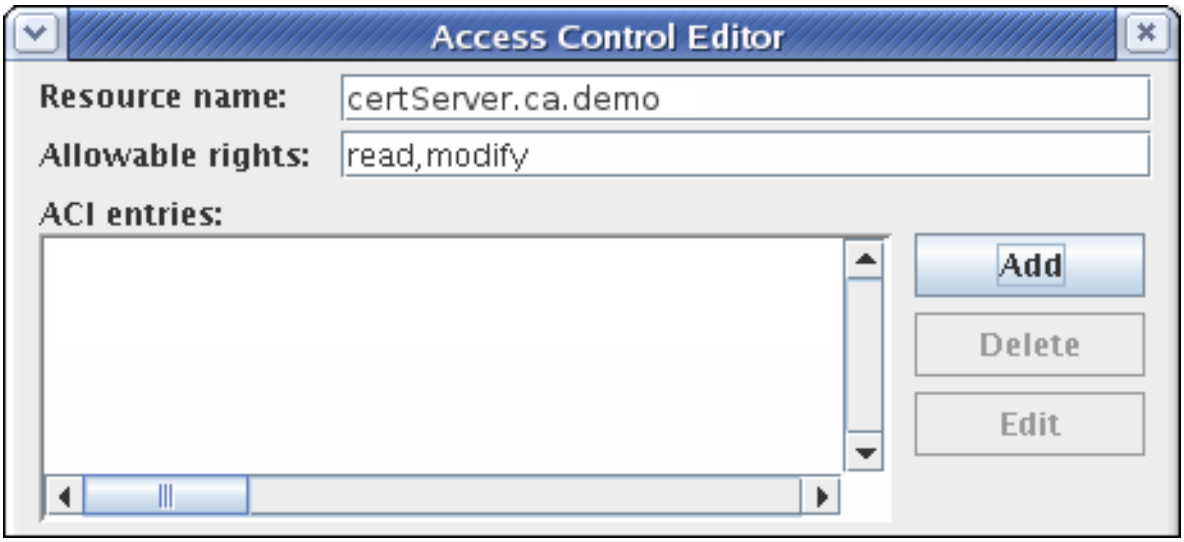
添加新 ACL：

1. 登录到管理控制台。
2. 选择 Access Control List。

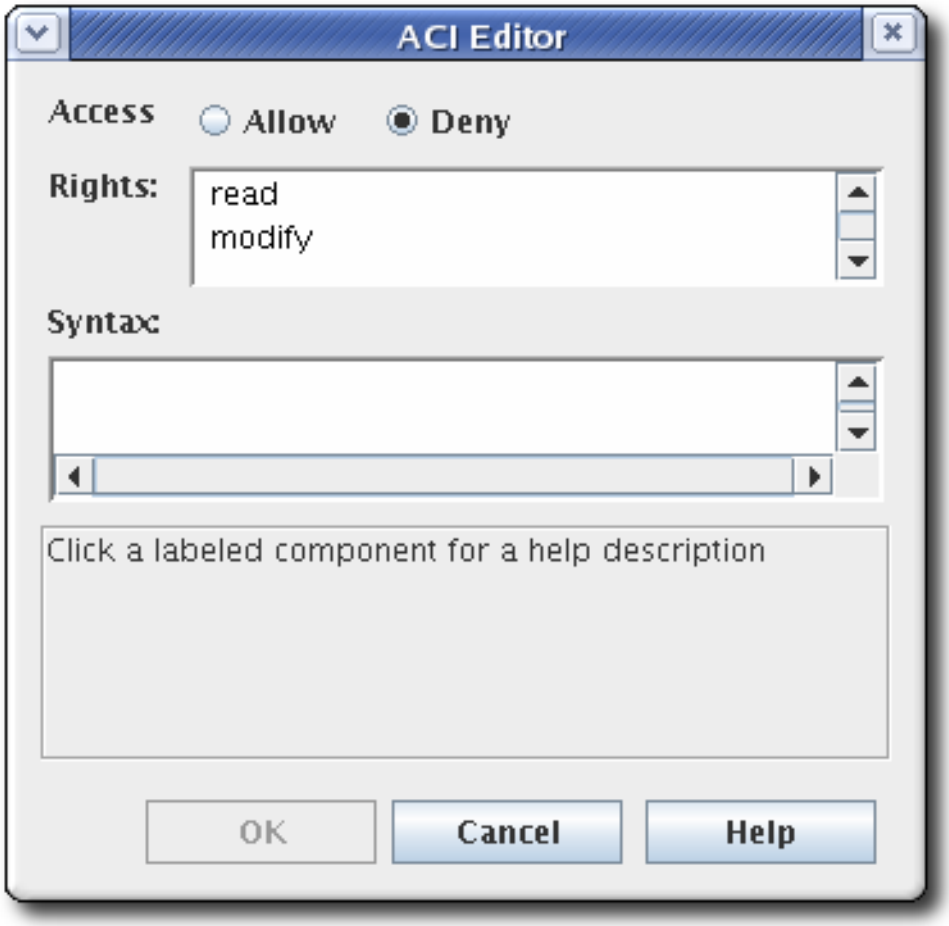


3. *单击 **Add** 以打开 **Access Control Editor**。*

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 allow 或 deny 单选按钮，以允许或拒绝指定组、用户或 IP 地址的操作。有关允许或拒绝访问的详情请参考 第 15.5.1 节 “关于访问控制”。

b. 设置权利。可用的选项有 读取 和修改。要同时选择，请在选择条目时保存 Ctrl 或 Shift 按钮。

c. 指定在 Syntax 字段中将被授予或拒绝访问权限的用户、组或 IP 地址。有关语法的详情，请查看 第 15.5.1 节 “关于访问控制”。

6. 单击 OK 以返回到 Access Control Editor 窗口。

7. 单击 OK 以存储 ACl。

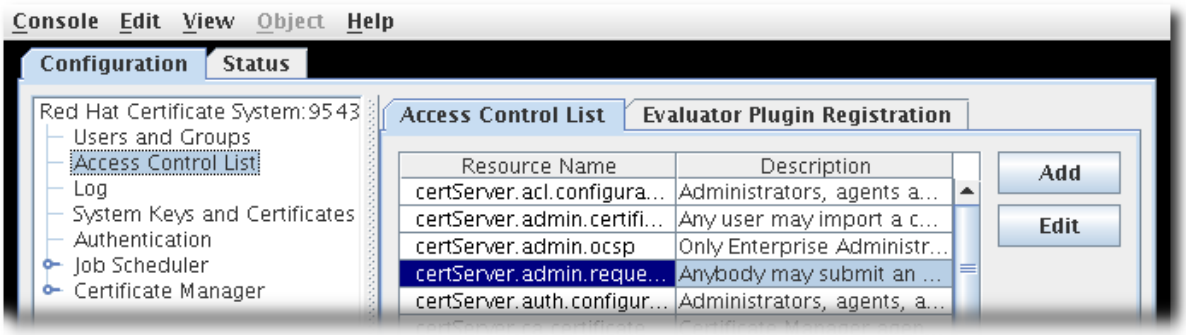
15.5.4. 编辑 ACL

ACL 存储在内部数据库中，且只能在管理控制台中进行修改。

编辑现有 ACL：

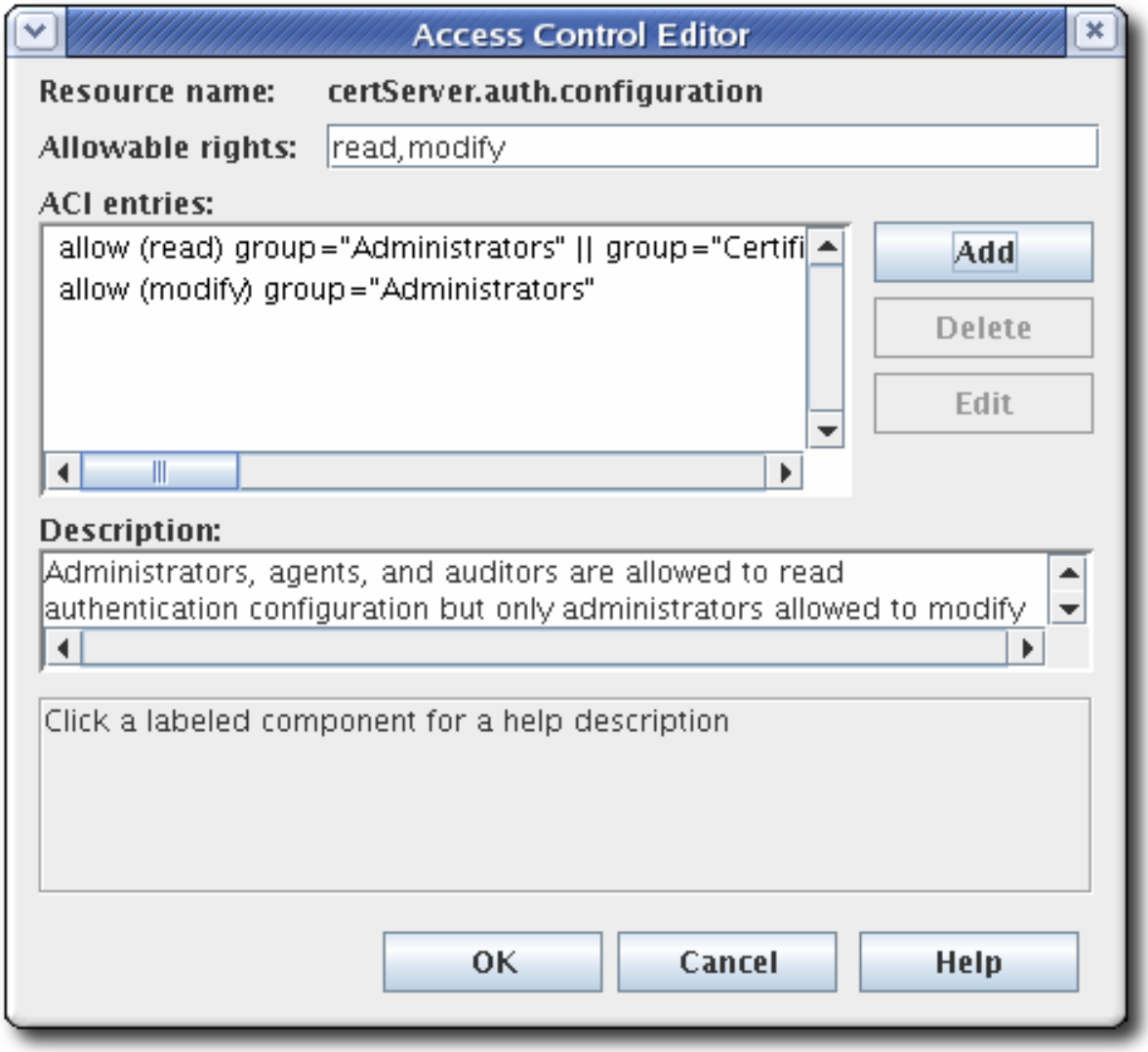
1. 登录到管理控制台。

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 Edit。

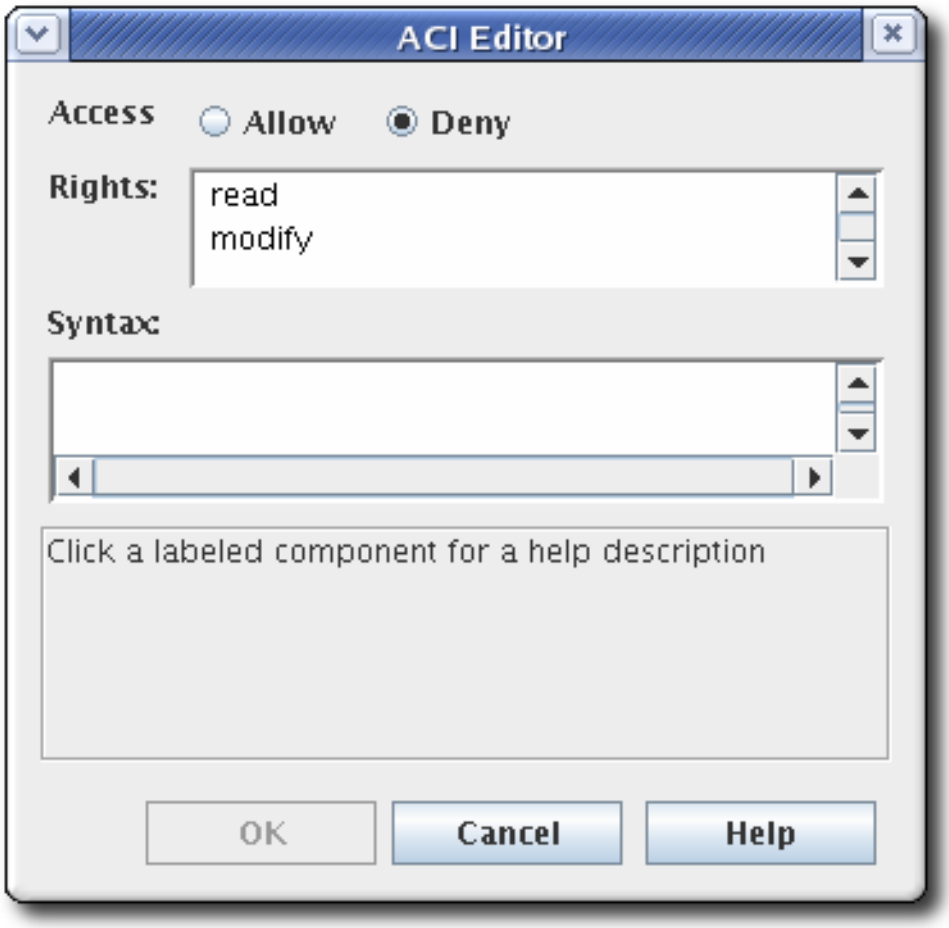
ACL 在 Access Control Editor 窗口中打开。



4.

要添加 ACI，请点 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 allow 或 deny 单选按钮，以允许或拒绝指定组、用户或 IP 地址的操作。有关允许或拒绝访问的详情请参考 第 15.5.1 节 “关于访问控制”。

b.

设置访问控制的权利。选项为 读取 和修改。要设置两者，请使用 Ctrl 或 Shift 按钮。

c.

指定在 Syntax 字段中将被授予或拒绝访问权限的用户、组或 IP 地址。有关语法的详情，请查看 第 15.5.1 节 “关于访问控制”。

第 16 章 配置子系统日志

证书系统子系统创建日志文件，记录与活动相关的事件，如管理、使用服务器支持的任何协议的通信，以及子系统使用的各种其他进程。在子系统实例运行时，它会在其管理的所有组件上保留信息和错误消息的日志。另外，Apache 和 Tomcat Web 服务器会生成错误和访问日志。

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。

日志插件模块是作为 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 pki_subsystem_log_path 中指定的任何目录中，而实例是使用 pkispawn 创建时在 pki_subsystem_log_path 中指定的任何目录中。 常规审计日志位于带有其他类型的日志的日志目录中，而签名的审计日志则写入 /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。

16.1. 关于证书系统日志

证书系统子系统保留几种不同类型的日志，它们根据子系统类型、服务类型和单个日志设置提供特定信息。可以为实例保留的日志类型取决于它所在的子系统类型。

16.1.1. 签名的审计日志

证书系统为所有事件维护审计日志，如请求、发布和撤销证书并发布 CRL。这些日志然后被签名。这允许检测到授权访问或活动。然后，外部审核员可以在需要时审核系统。分配的 auditor 用户帐户是唯一能够查看已签名的审计日志的帐户。此用户的证书用于签名和加密日志。审计日志记录配置为指定日志记录的事件。

签名的审计日志写入 /var/log/pki/instance_name/subsystem_name/signedAudit。但是，可以通过修改配置来更改日志的默认位置。

如需更多信息，请参阅 第 16.3.2 节 “使用签名的审计日志”。

16.1.2. 调试日志

调试日志（默认启用）会针对所有子系统进行维护，具有不同程度和信息类型。

调试日志包含子系统所执行的每个操作的非常具体的信息，包括运行、连接信息和服务器请求和响应消息的插件和 servlet。

第 16.2.1.1 节 “正在记录的服务” 中会简要讨论记录到调试日志的常规服务类型。这些服务包括授权请求、处理证书请求、证书状态检查以及归档和恢复密钥，以及访问 Web 服务。

CA、IADP、KRA 和 TKS 记录的调试日志记录有关子系统进程的详细信息。每个日志条目都有一个以下格式：

> [date:time] [processor]: servlet: message

消息 可以是子系统的返回消息，也可以是包含提交到子系统的值。

例如，TKS 记录了此消息用于连接 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 的，消息 是服务器有关 LDAP 连接的消息，且没有 servlet。

另一方面，CA 会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

在本例中，处理器 是 CA 代理端口上的 HTTP 协议，而它指定了 servlet 处理配置集，其中包含提供配置文件参数（请求的子系统所有者）及其值(KRA 启动请求)。

> 例 16.1. 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=MIIBozCCAZ8wggEFAgQqTfoHMIHHgAECpQ4wDDEKMAgGA1UEAxMBeKaBnzANBgkqhkiG9w0BAQEFAAOB...
> > [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=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxvkLP^*
M
HcN0cusY7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChV^M
k9HYDhmJ8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5I+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=MIICIKADAgECAgEAMA0GCSqGSIb3DQEBBQUAMEAxHjAcBgNVBAoTFVJlZGJ1ZGNv^M

bXB1dGVyIERvbWFpbjEeMBwGA1UEAxMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4X^M
DTA4MDYwNjE5NTkzOFoXDTA4MTIwMzE5NTkzOFowOzEhMB8GCSqGSIb3DQEJARYS^M
anNtaXRoQGV4YW1wbGUuY29tMRYwFAYKCZImiZPyLGQBARMGanNtaXRoMIGfMA0G^M
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxvkLPHcN0cusY^M
7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChVk9HYDhmJ^M
8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5I+2gE9PUznxJaMHTmlOqm4^M
HwFxzy0RRQIDAQABo4HFMIHCMB8GA1UdIwQYMBaAFG8gWeOJIMt+aO8VuQTMzPBU^M
78k8MEoGCCsGAQUFBwEBBD4wPDA6BggrBgEFBQcwAYYuaHR0cDovL3Rlc3Q0LnJl^M
ZGJ1ZGNvbXB1dGVyLmxvY2FsOjkwODAvY2Evb2NzcDAOBgNVHQ8BAf8EBAMCBeAw^M
HQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMCQGA1UdEQQdMBuBGSRyZXF1^M
ZXN0LnJlcXVlc3Rvcl9lbWFpbCQ=

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
```

16.1.2.1. 安装日志

所有子系统都会保留安装日志。

每次通过初始安装创建子系统时，或使用 **pkispawn** 创建额外的实例，它是安装含有完整调试输出的安装文件，包括任何错误，如果安装成功，则 **URL** 和 **PIN** 到实例的配置接口。该文件在实例的 **/var/log/pki/** 目录中创建，其名称格式为 **pki-subsystem_name-spawn.timestamp.log**。

安装日志中的每一行都遵循安装过程中的步骤。

例 16.2. CA 安装日志

```
...
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
```

16.1.2.2. Tomcat 错误和访问日志

CA、KRA、TKS、TKS 和 TPS 子系统将 Tomcat Web 服务器实例用于其代理和终端实体的接口。

错误和访问日志由 Tomcat Web 服务器创建，该服务器随证书系统一起安装并提供 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

这些日志在证书系统中不可用或可配置；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。

16.1.2.3. 自我测试日志

当服务器启动或手动运行时，在自tests 运行期间获取的自tests 日志消息。可以通过打开此日志来查看测试。此日志无法通过控制台进行配置，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 文件来配置日志的说明，请参阅 Red Hat Certificate System Planning、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此日志无关。有关 self-tests 的更多信息，请参阅 第 14.9 节 “运行自测

试”。

16.2. 管理日志

证书系统子系统日志文件记录了与该特定子系统实例中操作相关的事件。对于每个子系统，针对安装、访问和 Web 服务器等问题保留不同的日志。

所有子系统都有类似的日志配置、选项和管理路径。

16.2.1. 日志设置概述

配置日志的方式可能会影响证书系统性能。例如，日志文件轮转会变得太大，这会减慢子系统性能。本节介绍证书系统子系统记录的不同类型的日志，并涵盖日志文件轮转、缓冲日志记录和可用日志级别等重要概念。

16.2.1.1. 正在记录的服务

到日志文件的证书系统日志消息的所有主要组件和协议。表 16.1 “服务日志” 列出默认日志记录的服务。要查看特定服务记录的消息，请相应地自定义日志设置。详情请查看 第 16.3.1 节 “在控制台中查看日志”。

表 16.1. 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	记录与内部数据库相关的事件。

Service	描述
HTTP	记录与服务器 HTTP 活动相关的事件。请注意，HTTP 事件实际记录到属于证书系统附带的 Apache 服务器的错误日志，以提供 HTTP 服务。
密钥恢复授权	记录与 KRA 相关的事件。
LDAP	日志与 LDAP 目录相关的事件，用于发布证书和 CRL。
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，如命令行工具和其他进程。
请求队列	记录与请求队列活动相关的事件。
用户和组	记录与实例的用户和组相关的事件。

16.2.1.2. 日志级别(Message Categories)

证书系统服务记录的不同事件由日志级别决定，这有助于识别和过滤事件。表 16.2 “日志级别和清理日志消息” 中列出了不同的证书系统日志级别。

日志级别通过数字表示服务器要执行的日志级别的详细程度。

较高的优先级级别意味着较少的详细信息，因为仅记录高优先级的事件。

表 16.2. 日志级别和清理日志消息

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。

日志级别	消息类别	描述
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系统状态的一般信息，包括 证书系统初始化完成 和 请求成功操作的状态消息。
11-15	警告	这些消息只是警告，不表示服务器正常操作中的任何故障。
> 15	失败	这些消息指示阻止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 证书撤销）的故障，以及可能导致不可撤销的错误(服务器无法通过来自客户端的同一频道发送其处理的请求)。设置以上 15 的级别将最小化日志，因为仅记录失败。

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。

日志数据可能比较广泛，特别是在较低（更详细的）日志级别。确保主机机器对所有日志文件有足够的磁盘空间。务必要定义日志级别、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。

16.2.1.3. 缓冲和不缓冲的日志记录

Java 子系统支持对所有类型的日志进行缓冲的日志记录。可以为缓冲或未缓冲的日志配置服务器。

如果配置了缓冲的日志，服务器会为对应的日志创建缓冲区，并尽可能将消息保存到缓冲区中。只有在出现以下条件之一时，服务器才会将信息清除到日志文件：

- 缓冲区已满。当缓冲区大小等于或大于 bufferSize 配置参数指定的值时，缓冲区已满。此参数的默认值为 512 KB。
- 已达到缓冲区的冲刷间隔。当最后一次缓冲区刷新等于或大于 flushInterval 配置参数指定的值时，达到 flush 间隔。此参数的默认值为 5 秒。
- 从控制台读取当前日志时。服务器在查询当前日志时检索最新的日志。

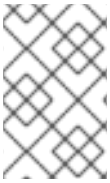
如果为未缓冲的日志配置了服务器，服务器会在生成到日志文件中时清除消息。由于服务器每次生成消息时执行 I/O 操作（写入日志文件），因此配置服务器以提高性能。

第 16.2.2 节 “在控制台中配置日志” 中描述了设置日志参数。

16.2.1.4. 日志文件轮转

子系统日志具有一个可选的日志设置，允许轮转并启动新的日志文件，而不是让日志文件无限期增长。当出现以下任一情况时，日志文件会被轮转：

- 达到相应文件的大小限制。对应的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。
- 达到相应文件的年龄限制。对应的日志文件等于或早于 rolloverInterval 配置参数指定的间隔。此参数的默认值为 2592000 秒（每 30 天）。

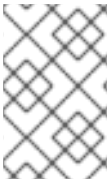


注意

将这些参数都设置为 0 可有效地禁用日志文件轮转。

轮转日志文件时，使用带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，表示轮转相应活跃日志文件的日期和时间。日期和时间具有 YYYYMMDD （年、月、天）和 HHMMSS （小时、分钟、秒）。

日志文件（特别是审计日志文件）包含重要信息。通过将整个 日志 目录复制到存档介质，应定期将这些文件归档到某些备份介质。



注意

证书系统不提供归档日志文件的任何工具或工具。

证书系统提供了一个命令行工具 signtool，它将在归档日志文件作为篡改检测方式签名。详情请查看 第 16.2.4.5 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自动签名的审计日志。有关签名审计日志的详情，请查看 第 16.2.4.3 节 “在控制台中配置签名的审计日志”。

轮转的日志文件不会被删除。

16.2.2. 在控制台中配置日志

日志可以通过子系统控制台和子系统的 CS.cfg 文件进行配置。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。

可以通过子系统控制台为 CA、OCSP、TKS 和 KRA 子系统配置审计日志。TPS 日志仅通过配置文件配置。

1. 在 Configuration 选项卡的导航树中，选择 Log。

2. Log Event Listener Management 选项卡列出了当前配置的监听程序。

   要创建新日志实例，请点 Add，然后从 Select Log Event Listener Plug-in Implementation 窗口中的列表中选择模块插件。

3. 设置或修改 Log Event Listener Editor 窗口中的字段。不同的参数列在 表 16.3 “日志事件 Listener 字段” 中。

表 16.3. 日志事件 Listener 字段

字段	描述
日志事件 Listener ID	提供标识监听程序的唯一名称。名称可以有字母的任意组合(A 到 zZ)、数字(0 到 9)、下划线(_)和连字符(-)，但它不能包含其他字符或空格。
type	提供日志文件的类型。事务 记录审计日志。
enabled	设置日志是否活跃。只有启用的日志才会记录事件。该值可以是 true 或 false。

字段	描述
level	在文本字段中设置日志级别。必须在字段中输入该级别，没有选择菜单。选择是 Debug、info、warning、Failed、Misconfiguration、Catastrophe 和 Security。 如需更多信息，请参阅 第 16.2.1.2 节 “日志级别(Message Categories)”。
fileName	将完整路径（包括文件名）提供给日志文件。子系统用户应具有文件的读/写权限。
bufferSize	为日志设置缓冲区大小(KB)。缓冲区达到这个大小后，缓冲区的内容将被清除并复制到日志文件中。默认大小为 512 KB。有关缓冲的日志的详情，请参考 第 16.2.1.3 节 “缓冲和不缓冲的日志记录”。
flushInterval	设置缓冲区内容被清除并添加到日志文件中的时间。默认间隔为 5 秒。
maxFileSize	以 KB 为单位设置日志文件的大小，然后再轮转日志文件。达到这个大小后，文件将复制到轮转的文件，日志文件将启动新的文件。有关日志文件轮转的详情，请参考 第 16.2.1.4 节 “日志文件轮转”。默认大小为 2000 KB。
rolloverInterval	设置服务器轮转活跃日志文件的频率。可用选项包括每小时、每天、每周、每月和年。默认值为 monthly。如需更多信息，请参阅 第 16.2.1.4 节 “日志文件轮转”。

16.2.3. 在 CS.cfg 文件中配置日志

有关如何通过编辑 CS.cfg 文件来配置日志的说明，请参阅 Red Hat Certificate System Planning、安装和部署指南中的 CS.cfg 文件中的配置日志 部分。

16.2.4. 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 属性设置为 true，则审计日志会使用属于服务器的日志签名证书进行签名。审核员可以使用此证书来验证日志没有被篡改。

默认情况下，常规审计日志位于带有其他类型的日志的 /var/log/pki/instance_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。

签名的审计日志会创建一个日志记录系统事件，并从潜在的事件列表中选择事件。启用后，签名的审计日志会记录一组有关所选事件活动的详细信息。

当实例首次创建时，会默认配置签名的审计日志，但可以在安装后配置签名的审计日志。（请参阅 第 16.2.4.2 节 “安装后启用签名的审计日志”。） 也可以在配置后编辑配置或更改签名证书，如 第 16.2.4.3 节 “在控制台中配置签名的审计日志” 所述。

16.2.4.1. 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。

16.2.4.2. 安装后启用签名的审计日志

当实例首次使用 pki_audit_group 部署参数和 pkispawn 命令时，可以默认启用签名的审计日志。但是，如果在创建实例时没有配置签名的审计日志，可通过将审计日志目录的所有权重新分配给审核员系统用户组来启用它们。

1. 停止实例：

```
# pki-server stop instance_name
```

2. 将签名的审计日志目录的组所有权设置为 PKI 审核员操作系统组，如 pkiaudit。这允许 PKI 审核员组中的用户具有对 signedAudit 目录所需的读取访问权限，以验证日志文件上的签名。除了证书系统用户帐户 pkiuser外，用户没有对这个目录中日志文件的写入权限。

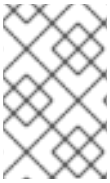
```
chgrp -R pkiaudit /var/log/pki/instance_name/subsystem_name/signedAudit
```

3. 重启实例：

```
# pki-server start instance_name
```

16.2.4.3. 在控制台中配置签名的审计日志

当实例首次创建时，会默认配置签名的审计日志，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，因为它们可能较大。

通过设置 logSigning 参数 来启用 并提供用于签署日志的证书 nickname，将日志设置为签名的审计日志。在第一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 工具来验证签名的审计日志没有被篡改。

在配置了子系统时，会创建并启用签名的审计日志，但需要额外的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 文件来创建和配置审计日志，请参阅 Red Hat Certificate System Planning, Installation, and Deployment Guide 中的 CS.cfg 文件中的配置日志。

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. 点 Edit/View。

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。在配置了子系统时会创建一个审计签名证书，它有一个别名，如 auditSigningCert cert-instance_name subsystem_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中的证书。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** 以启用签名的日志记录。

- 设置记录到审计日志的任何事件。附录 E, 审计事件 列出可日志的事件。日志事件用逗号分开，没有空格。

6. 为日志设置任何其他设置，如文件名、日志级别、文件大小或轮转计划。

注意

默认情况下，常规审计日志位于带有其他类型的日志的 **/var/log/pki/instance_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，请查看 第 15.3.2.1 节 “创建用户”。

审核员可以使用 **audit Verify** 工具验证 日志。有关使用 此工具的详情，请查看 **auditVerify (1)** 手册页。

16.2.4.4. 处理审计日志失败

有些事件可能会导致审计日志功能失败，因此无法将事件写入日志中。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更改时，审计日志记录可能会失败。如果审计日志记录失败，证书系统实例将以以下方式关闭。

- servlet 被禁用，也不会处理新的请求。
- 所有待处理的和新请求都会被终止。
- 子系统已关闭。

发生这种情况时，管理员和审核员应当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 问题得到解决时，审核员应确保最后的审计日志条目已被签名。如果没有，则应手动签名 (第 16.2.4.5 节 “签名日志文件”)、存档和删除来保留它们，以防止以后进行审核验证失败。完成此操作后，管理员可以重新启动证书系统。

16.2.4.5. 签名日志文件

证书系统可以在归档或分发用于审计目的之前数字签名日志文件。此功能允许检查文件以篡改。

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；此工具手动签署归档的日志。有关签名审计日志的详情，请查看 第 16.2.4.3 节 “在控制台中配置签名的审计日志”。

对于签名日志文件，请使用名为 Signing Tool (signtool)的命令行工具。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。

作为具有审核员权限的用户，使用 signtool 命令为日志目录签名：

```
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的路径。
- cert_nickname 指定用于签名的证书的别名。

- output 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。

16.2.4.6. 过滤审计事件

在证书系统管理员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 过滤器相同。但是，证书系统只支持以下过滤器：

表 16.4. 支持的审计事件过滤器

类型	æ ¼å¼	示例
存在	(attribute=*)	(ReqID=*)
ç›¸ç‰	(attribute=value)	(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，请查看 红帽目录服务器管理指南中的使用 复合搜索过滤器。

> 例 16.3. 过滤审计事件
>
> 为配置文件证书请求记录失败的事件，以及将 InfoName 字段设置为 rejectReadon 或 cancelReason 的处理证书请求的事件：
>
> 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：

```
# pki-server restart instance_name
```

16.2.5. 管理日志模块

允许的日志类型及其行为是通过 **log** 模块 插件配置的。可以创建新的日志记录模块，并用于编写自定义日志。

新的日志插件模块可以通过控制台注册。注册新模块涉及指定模块名称和实施日志接口的 **Java™** 类的完整名称。

在注册插件模块前，请将模块的 **Java™** 类放在 类 目录中；实施必须位于类路径上。

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** 目录中创建一个目录来保存自定义类，以便 **CA** 可以访问它们。

```
mkdir /var/lib/pki/pki-tomcat/webapps/ca/WEB-INF/classes
```

4. 将所有者设置为证书系统用户**(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 插件注册 选项卡。

   c. 点击 Register。

      此时会出现 Register Log Event Listener 插件实施 窗口。

   d. 为插件模块和 Java™ 类名称指定名称。

      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 的软件包中注册名为 customLog 的类，类名称为 com.customplugins.customLog。

   e. 点确定。

可以通过控制台删除不需要的日志插件模块。在删除模块前，请先删除所有基于这个模块的监听程序；请参阅 第 16.2.1.4 节 “日志文件轮转”。

16.3. 使用日志

16.3.1. 在控制台中查看日志

若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。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，仅具有审核员角色的用户才可以使用 第 16.3.2 节 “使用签名的审计日志” 中详述的方法访问审计日志。

查看日志文件的内容：

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。

   - entry - 要显示的最大条目数。达到这个限制时，证书系统会返回与搜索请求匹配的任何条目。零(0)表示没有返回任何消息。如果该字段为空，服务器会返回每个匹配的条目，而不考虑找到的数字。

   - Source - 选择要显示日志消息的证书系统组件或服务。选择 All 表示日志记录到此文件的所有组件的消息都会被显示。

   - level - 选择一个消息类别，它代表过滤消息的日志级别。

   - filename - 选择要查看的日志文件。

5. 单击 Refresh。

6. 要查看完整的条目，请双击该条目，或者选择条目，然后单击 View。

16.3.2. 使用签名的审计日志

本节介绍审核员组中的用户如何显示并验证签名的审计日志。

16.3.2.1. 列出审计日志

以具有审核员的用户身份，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，要列出托管在 server.example.com 上的 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
----------------------------
```

命令使用带有存储在 ~/.dogtag/nssdb/ 目录中的 auditor nickname 的客户端证书，以向 CA 进行身份验证。有关命令中使用的参数和替代验证方法的详情，请查看 pki(1) man page。

16.3.2.2. 下载审计日志

以具有审核员的用户身份，使用 pki subsystem-audit-file-retrieve 命令从服务器下载特定的审计日志。

例如，要从托管在 server.example.com 上的 CA 下载审计日志文件：

1. （可选）列出 CA 上的可用日志文件。请参阅 第 16.3.2.1 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
   ```

   命令使用带有存储在 ~/.dogtag/nssdb/ 目录中的 auditor nickname 的客户端证书，以向 CA 进行身份验证。有关命令中使用的参数和替代验证方法的详情，请查看 pki(1) man page。

下载日志文件后，您可以使用 grep 工具搜索特定的日志条目：

```
# grep "\[AuditEvent=ACCESS_SESSION_ESTABLISH\]" log_file
```

16.3.2.3. 验证签名的审计日志

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 Planning, Installation, installation, Installing, deployment Guide 中的 第 2.5.1.1 节 “pki CLI 初始化” 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，请导入它：

   a. 为您要验证的子系统日志搜索审计签名证书。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：

   ```
   # 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 16.3.2.2 节 “下载审计日志”。

4. 验证审计日志。

a. 创建一个包含您要按时间顺序验证的审计日志文件列表的文本文件。例如：

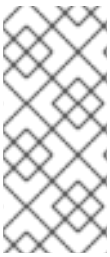
```
# cat > ~/audit.txt << EOF
ca_audit.20170331225716
ca_audit.20170401001030
ca_audit
EOF
```

b. 使用 auditVerify 工具验证签名。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 审计验证 的详情，请查看 AuditVerify(1) man page。

16.3.3. 显示操作系统级审计日志



注意

要使用以下说明来查看操作系统级别的审计日志，必须根据 Red Hat Certificate System 规划、安装和部署指南中的 启用操作系统级审计日志 部分配置 auditd 日志记录框架。

要显示操作系统级访问日志，以 root 用户身份使用 ausearch 工具，或使用 sudo 工具以特权用户身份使用 ausearch 工具。

16.3.3.1. 显示审计日志删除事件

由于这些事件是密钥的（使用 rhcs_audit_deletion），因此请使用 -k 参数来查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_deletion
```

16.3.3.2. 显示对 Secret 和私钥的 NSS 数据库的访问

由于这些事件是密钥的（使用 rhcs_audit_nssdb），因此请使用 -k 参数来查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_nssdb
```

16.3.3.3. 显示时间更改事件

由于这些事件是密钥的（使用 rhcs_audit_time_change），因此请使用 -k 参数来查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_time_change
```

16.3.3.4. 显示软件包更新事件

由于这些事件是类型的消息（类型为 SOFTWARE_UPDATE），因此请使用 -m 参数来查找与类型匹配的事件：

```
# ausearch -m SOFTWARE_UPDATE
```

16.3.3.5. 显示 PKI 配置的更改

由于这些事件是密钥的（使用 rhcs_audit_config），因此请使用 -k 参数来查找与该密钥匹配的事件：

```
# ausearch -k rhcs_audit_config
```

16.3.4. 智能卡错误代码

智能卡可向 TPS 报告某些错误代码；这些错误代码记录在 TPS 的调试日志中，具体取决于消息的原因。

表 16.5. 智能卡错误代码

返回码	描述
常规错误代码	
6400	没有特定的诊断
6700	Lc 中错误的长度

返回码	描述
6982	未满足安全性状态
6985	不满意的使用条件
6a86	P1 P2 不正确
6d00	无效的指令
6e00	无效的类
安装负载错误	
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	应用程序已逻辑删除
6581	内存故障
6985	无法删除引用的数据
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
获取数据错误	
6a88	未找到引用的数据
获取状态错误	

返回码	描述
6310	更多可用数据
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 不正确
6985	不满意的使用条件

第 17 章 管理子系统证书

本章概述了使用证书：可以使用哪些类型和格式，如何通过 HTML 端点表单和证书系统控制台来请求和创建它们，以及如何在证书系统和不同的客户端中安装证书。另外，还有有关通过控制台管理证书的信息，并将服务器配置为使用它们。

17.1. 所需的子系统证书

每个子系统都有一组定义的证书，这些证书必须发布到子系统实例，才能执行操作。证书管理器配置期间设置的证书内容有某些细节，但根据证书类型，限制、设置和属性的不同注意事项；规划 证书系统规划、安装和部署指南中的证书格式。

17.1.1. 证书管理器证书

安装证书管理器后，会生成 CA 签名证书、SSL 服务器证书和 OCSP 签名证书的密钥和请求。证书在配置完成之前创建。

CA 证书请求作为自签名请求提交给 CA，然后签发证书并完成创建自签名 root CA，或者提交到第三方公共 CA 或其他证书系统 CA。当外部 CA 返回证书时，证书会被安装，并且完成从属 CA 的安装。

- 第 17.1.1.1 节 “CA 签名密钥和证书”
- 第 17.1.1.2 节 “OCSP 签名密钥和证书”
- 第 17.1.1.3 节 “子系统证书”
- 第 17.1.1.4 节 “SSL 服务器密钥检测和证书”
- 第 17.1.1.5 节 “审计日志签名密钥和证书”

17.1.1.1. CA 签名密钥和证书

每个证书管理器都有一个 CA 签名证书，其与证书管理器用来为证书签名的私钥对应的公钥对应。在安装证书管理器时，会创建并安装此证书。证书的默认别名是 caSigningCert cert-instance_ID CA，其

中 instance_ID 标识证书管理器实例。证书的默认有效期为五年。

CA 签名证书的主题名称反映了在安装过程中设置的 CA 的名称。证书管理器签名或签发的所有证书都包含此名称来标识证书的签发者。

证书管理器的状态是 root 或从属 CA 的状态，它由其 CA 签名证书是自签名还是由另一个 CA 签名，这会影响证书的主题名称。

- 如果证书管理器是 root CA，则其 CA 签名证书是自签名的，这意味着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，其 CA 签名证书由另一个 CA 签名，通常是 CA 层次结构上面的级别（可能或可能不是 root CA）。在证书管理器可用于向它们发布证书之前，必须将 root CA 的签名证书导入到单独的客户端和服务器中。

注意

CA 名称 不能更改，或者所有之前发布的证书都无效。同样，使用新密钥对恢复 CA 签名证书会导致旧密钥对的所有证书无效。

17.1.1.2. OCSP 签名密钥和证书

OCSP 签名证书的主题名称的格式是 cn=OCSP cert-instance_ID CA，它包含扩展，如 OCSPSigning 和 OCSPNoCheck，用于签名 OCSP 响应。

OCSP 签名证书的默认别名为 ocspSigningCert cert-instance_ID，其中 instance_ID CA 标识证书管理器实例。

证书管理器使用与 OCSP 签名证书的公钥对应的 OCSP 私钥，在查询证书撤销状态时，向 OCSP 兼容客户端签名 OCSP 响应。

17.1.1.3. 子系统证书

安全域的每个成员都会发布一个服务器证书，用于与其他域成员之间的通信，后者与服务器 SSL 证书分开。此证书由安全域 CA 签名；对于安全域 CA 本身，其子系统证书由自身签名。

证书的默认别名是 subsystemCert cert-instance_ID。

17.1.1.4. SSL 服务器密钥检测和证书

每个证书管理器至少有一个 SSL 服务器证书是在安装证书管理器时首次生成的。证书的默认别名是 Server-Cert cert-instance_ID，其中 instance_ID 标识证书管理器实例。

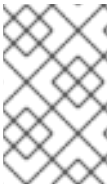
默认情况下，证书管理器使用单个 SSL 服务器证书进行身份验证。但是，可以请求额外的服务器证书来用于不同的操作，如将证书管理器配置为使用单独的服务器证书对最终用户服务接口和代理服务接口进行身份验证。

如果为启用了 SSL 的发布目录通信配置了证书管理器，它将默认使用其 SSL 服务器证书进行客户端身份验证到发布目录。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。

17.1.1.5. 审计日志签名密钥和证书

CA 保留服务器上发生的所有事件的安全审计日志。为确保审计日志未被篡改，日志文件由特殊的日志签名证书签名。

在第一次配置服务器时，会发布审计日志签名证书。



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。

17.1.2. 在线证书状态管理器证书

首次配置在线证书状态管理器后，会创建所有必需的证书的密钥，以及 OCSP 签名、SSL 服务器、审计日志签名和子系统证书的证书请求。这些证书请求提交到 CA （证书系统 CA 或第三方 CA），且必须安装在在线证书状态管理器数据库中来完成配置过程。

- 第 17.1.2.2 节 “SSL 服务器密钥检测和证书”
- 第 17.1.2.3 节 “子系统证书”

- 第 17.1.2.4 节 “审计日志签名密钥和证书”
- 第 17.1.2.5 节 “识别在线证书状态管理器证书”

17.1.2.1. OCSP 签名密钥和证书

每个在线证书状态管理器都有一个证书 OCSP 签名证书，该证书与在线证书状态管理器用来签署 OCSP 响应的私钥对应。在线证书状态管理器的签名提供了在线证书状态管理器处理请求的持久概念验证。当配置了在线证书状态管理器时，会生成此证书。证书的默认别名为 ocspSigningCert cert-instance_ID，其中 instance_ID OSCP 是在线证书状态管理器实例名称。

17.1.2.2. SSL 服务器密钥检测和证书

每个在线证书状态管理器至少有一个 SSL 服务器证书，该证书是在配置在线证书状态管理器时生成的。证书的默认别名是 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 证书状态管理器将其服务器证书用于在线证书状态管理器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身份验证。可以安装其他服务器证书并用于不同的目的。

17.1.2.3. 子系统证书

安全域的每个成员都会发布一个服务器证书，用于与其他域成员之间的通信，后者与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认别名是 subsystemCert cert-instance_ID。

17.1.2.4. 审计日志签名密钥和证书

OCSP 保留服务器上发生的所有事件的安全审计日志。为确保审计日志未被篡改，日志文件由特殊的日志签名证书签名。

在第一次配置服务器时，会发布审计日志签名证书。

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。

17.1.2.5. 识别在线证书状态管理器证书

根据签署在线证书状态管理器的 SSL 服务器证书的 CA，可能需要获取证书并发布证书管理器识别的 CA。

- 如果在线证书状态管理器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，则证书链会在配置在线证书状态管理器时导入并标记。不需要其他配置。但是，如果服务器证书由外部 CA 签名，则必须导入证书链才能完成配置。

注意

在配置时 OCSP Manager 会自动信任安全域中的每个 CA。CA 面板中配置的 CA 证书链中的每个 CA 都是，但由 OCSP Manager 自动信任。安全域中的其他 CA，但不能在证书链中手动添加。

17.1.3. 密钥恢复授权证书

KRA 使用以下密钥对和证书：

- 第 17.1.3.1 节 “传输密钥描述和证书”
- 第 17.1.3.2 节 “存储密钥分区”

- 第 17.1.3.3 节 “SSL 服务器证书”
- 第 17.1.3.4 节 “子系统证书”
- 第 17.1.3.5 节 “审计日志签名密钥和证书”

17.1.3.1. 传输密钥描述和证书

每个 KRA 都有一个传输证书。用于生成传输证书的密钥对的公钥供客户端软件用来加密最终实体的私钥，然后再将其发送到存档的 KRA；只有能够生成双密钥对的客户端使用传输证书。

17.1.3.2. 存储密钥分区

每个 KRA 都有存储密钥对。 KRA 使用此密钥对的公钥组件在归档密钥时加密（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。有关如何使用这个密钥对的详情，请参考 第 4 章 设置 Key Archival 和恢复。

使用存储密钥加密的密钥只能由授权密钥恢复代理来检索。

17.1.3.3. SSL 服务器证书

每个证书系统 KRA 至少有一个 SSL 服务器证书。配置 KRA 时会生成第一个 SSL 服务器证书。证书的默认别名是 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的 CA 发布，可以是证书系统 CA 或第三方 CA。要查看签发者名称，请在 KRA 控制台中的 System Keys 和 Certificates 选项中打开证书详情。

KRA 使用其 SSL 服务器证书对 KRA 代理服务接口进行服务器端身份验证。默认情况下，密钥恢复授权使用单个 SSL 服务器证书进行身份验证。但是，可以为 KRA 请求并安装额外的 SSL 服务器证书。

17.1.3.4. 子系统证书

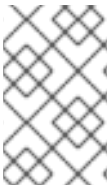
安全域的每个成员都会发布一个服务器证书，用于与其他域成员之间的通信，后者与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认别名是 subsystemCert cert-instance_ID。

17.1.3.5. 审计日志签名密钥和证书

KRA 保留服务器上发生的所有事件的安全审计日志。为确保审计日志未被篡改，日志文件由特殊的日志签名证书签名。

在第一次配置服务器时，会发布审计日志签名证书。



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。

17.1.4. 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 17.1.4.1 节 “SSL 服务器证书”
- 第 17.1.4.2 节 “子系统证书”
- 第 17.1.4.3 节 “审计日志签名密钥和证书”

17.1.4.1. SSL 服务器证书

每个证书系统 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。证书的默认别名是 Server-Cert cert-instance_ID。

17.1.4.2. 子系统证书

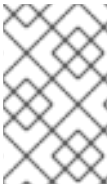
安全域的每个成员都会发布一个服务器证书，用于与其他域成员之间的通信，后者与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认别名是 subsystemCert cert-instance_ID。

17.1.4.3. 审计日志签名密钥和证书

TKS 保留服务器上发生的所有事件的安全审计日志。为确保审计日志未被篡改，日志文件由特殊的日志签名证书签名。

在第一次配置服务器时，会发布审计日志签名证书。



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。

17.1.5. 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 17.1.5.1 节 “SSL 服务器证书”
- 第 17.1.5.2 节 “子系统证书”
- 第 17.1.5.3 节 “审计日志签名密钥和证书”

17.1.5.1. SSL 服务器证书

每个证书系统 TPS 至少有一个 SSL 服务器证书。配置 TPS 时会生成第一个 SSL 服务器证书。证书的默认别名是 Server-Cert cert-instance_ID。

17.1.5.2. 子系统证书

安全域的每个成员都会发布一个服务器证书，用于与其他域成员之间的通信，后者与服务器 SSL 证书分开。此证书由安全域 CA 签名。

证书的默认别名是 subsystemCert cert-instance_ID。

17.1.5.3. 审计日志签名密钥和证书

TPS 保留服务器上发生的所有事件的安全审计日志。为确保审计日志未被篡改，日志文件由特殊的日志签名证书签名。

在第一次配置服务器时，会发布审计日志签名证书。

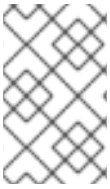
17.1.6. 关于子系统证书密钥类型

当您创建新实例时，您可以在传递给 pkispawn 工具的配置文件中指定密钥类型和密钥大小。

例 17.1. CA 的密钥类型相关配置参数

以下是与类型相关的关键参数，包括示例值。您可以在创建新 CA 时传递给 pkispawn 的配置文件中设置这些参数。

```
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* 一节。
- 有关参数和示例的描述的 pki_default.cfg(5) man page。

17.1.7. 使用 HSM 存储子系统证书

默认情况下，密钥和证书分别存储在本地管理的数据库、key4.db 和 cert9.db 中，位于 /var/lib/pki/instance_name/alias 目录中。但是，红帽认证系统还支持硬件安全模块(HSM)，外部设备可将密钥和证书存储在网络上的集中位置。使用 HSM 可使一些功能（如克隆）变得更加容易，因为实例的密钥和证书可以被只读访问。

当使用 HSM 来存储证书时，则 HSM 名称被放在证书 nickname 前，并在子系统配置中使用全名，如 server.xml 文件。例如：

```
serverCert="nethsm:Server-Cert cert-instance_ID
```

注意

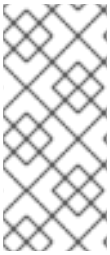
单个 HSM 可用于存储 mulitple 子系统实例的证书和密钥，这些实例可能安装到多个主机上。使用 HSM 时，子系统的任何证书 nickname 都必须对 HSM 上管理的每个子系统实例都是唯一的。

证书系统支持两种类型的 HSM、nCipher netHSM 和 Chrysalis LunaSA。

17.2. 通过控制台请求证书

CA、IADP、KRA 和 TKS 的证书设置向导自动执行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。

17.2.1. 请求签名证书



注意

用户从稍后使用的计算机生成并提交客户端请求非常重要，因为请求进程的一部分在本地计算机上生成私钥。如果需要位置 independence，请使用硬件令牌（如智能卡）来存储密钥对和证书。

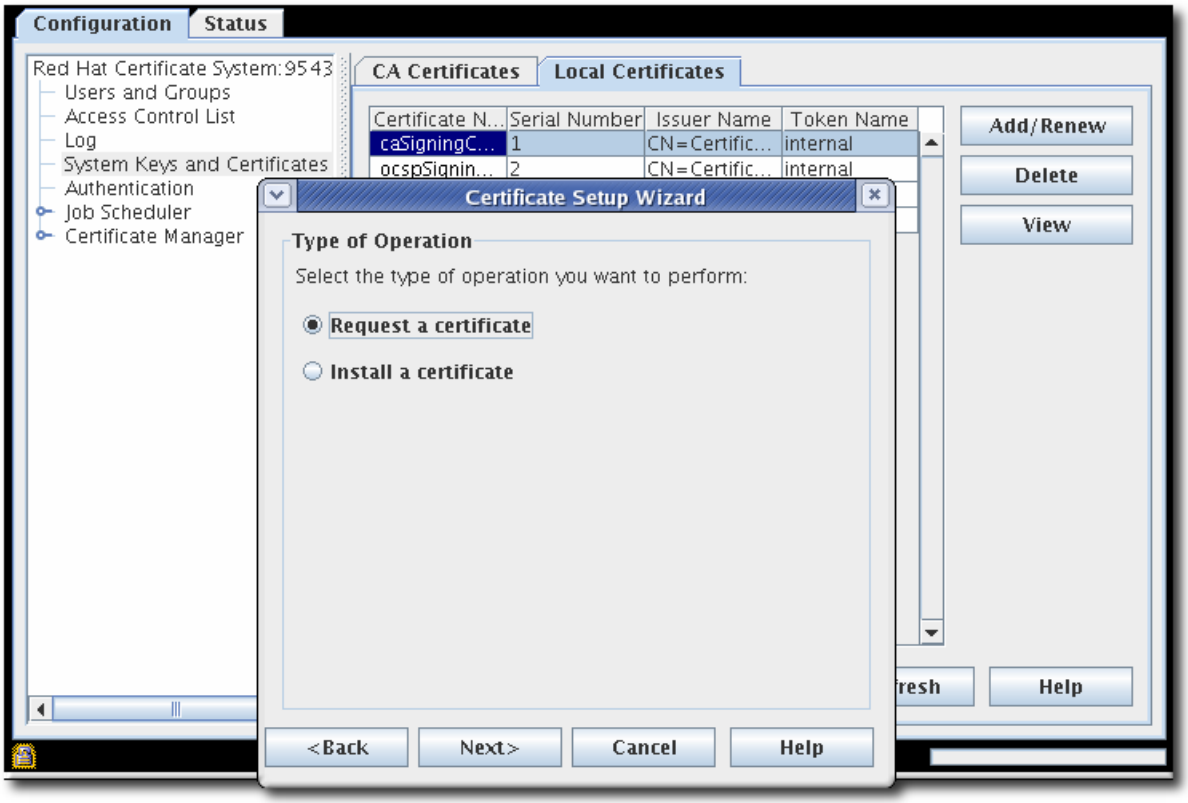
1. 打开子系统控制台。例如：

```
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 Local Certificates 选项卡。

4. 单击 Add/Renew。



5. 选择 Request a certificate 单选按钮。

6.

选择要请求的签名证书类型。

7.

选择哪个类型的 CA 将为请求签名，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，也可以是列出的外部令牌之一。

若要创建新证书，您必须创建一个新密钥对。使用现有密钥对将只续订现有证书。

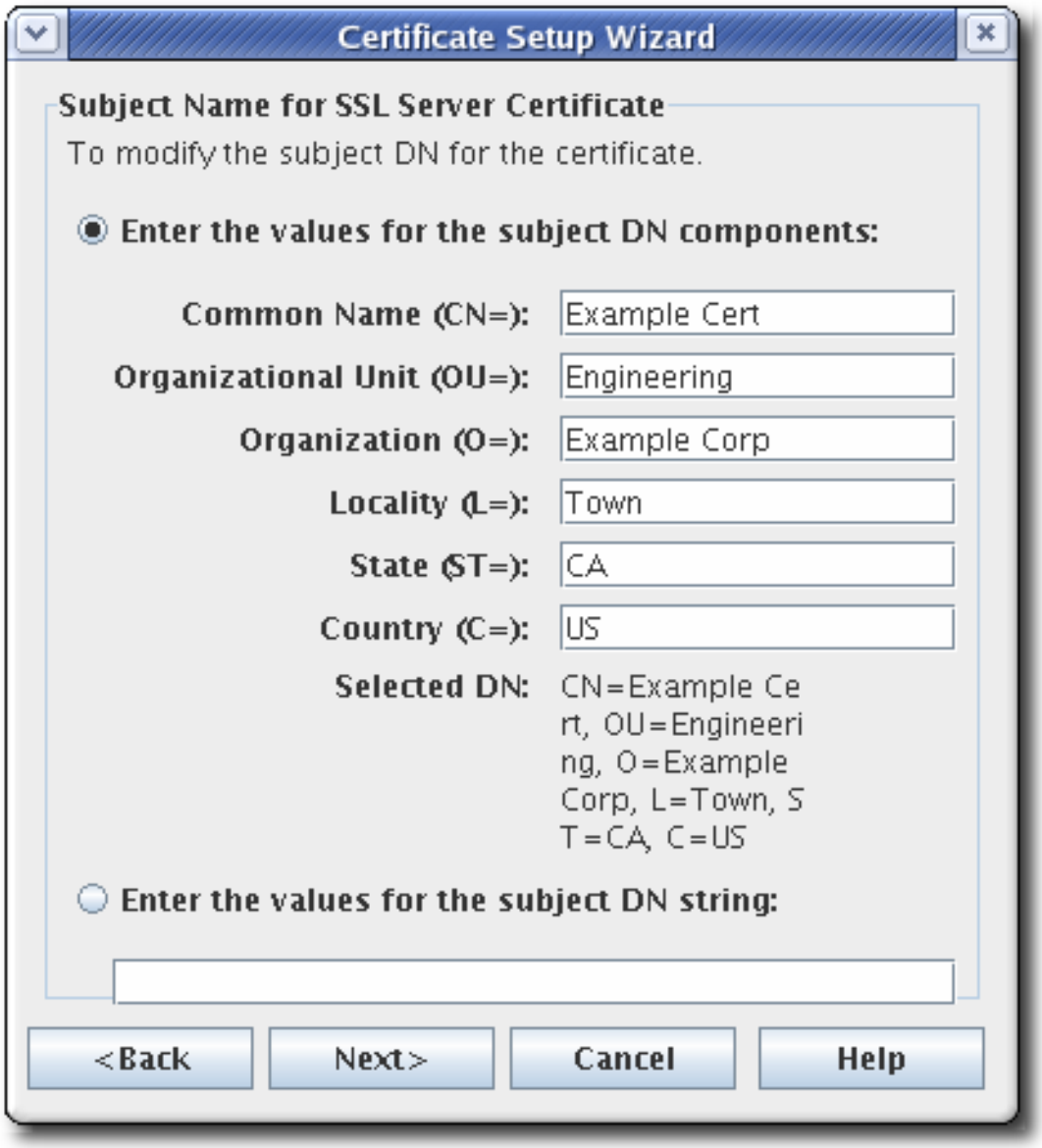

Certificate Setup Wizard
Key-Pair Information for the SSL Server Certificate
Select the token (cryptographic device) on which to generate the key pair:
Token:
internal
Specify which key pair to use for this request:
Use existing key pair
Create new key pair
Specify the key type and key length:
Key type:
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.

指定主题名称。为单个 DN 属性输入值来构建主题 DN 或输入完整的字符串。

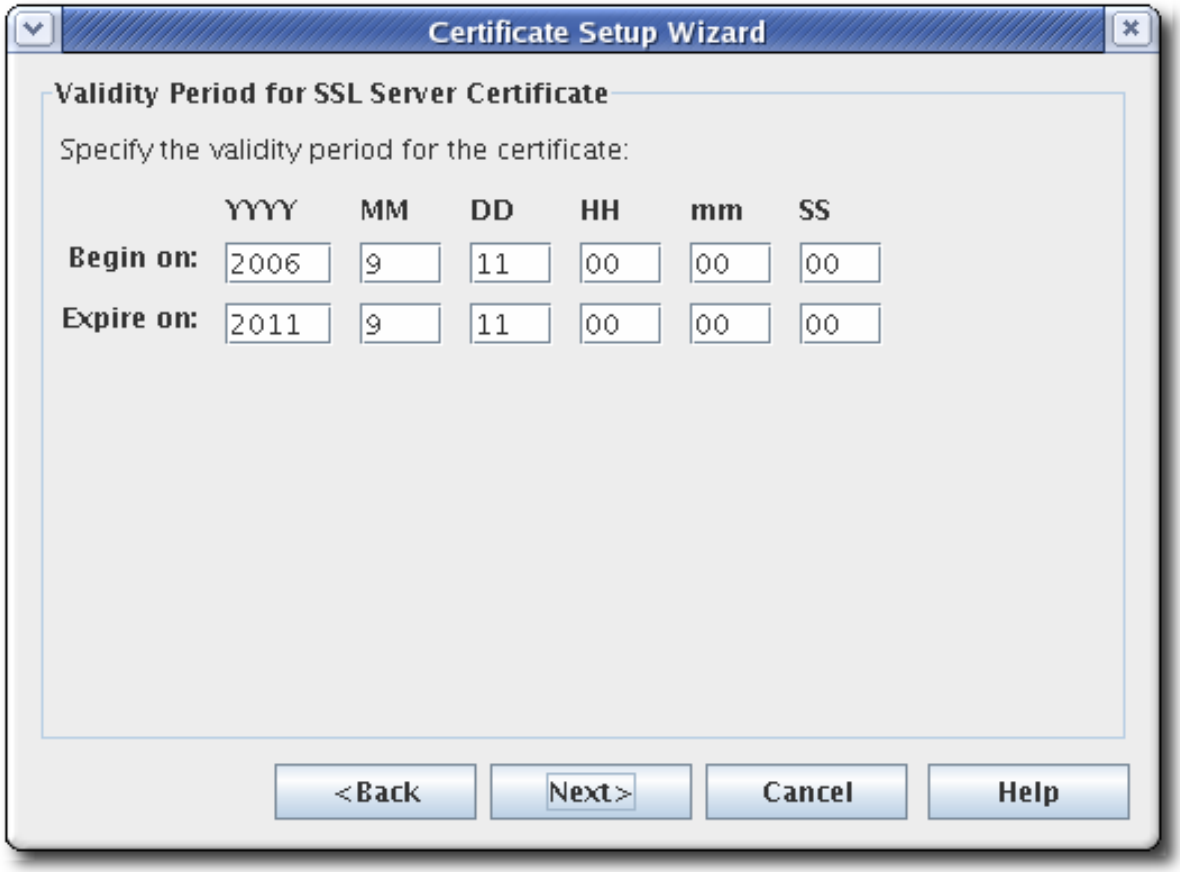


证书请求表单支持通用名称、机构单元和请求名称字段的所有 UTF-8 字符。

这个支持不包括支持国际化域名。

11.

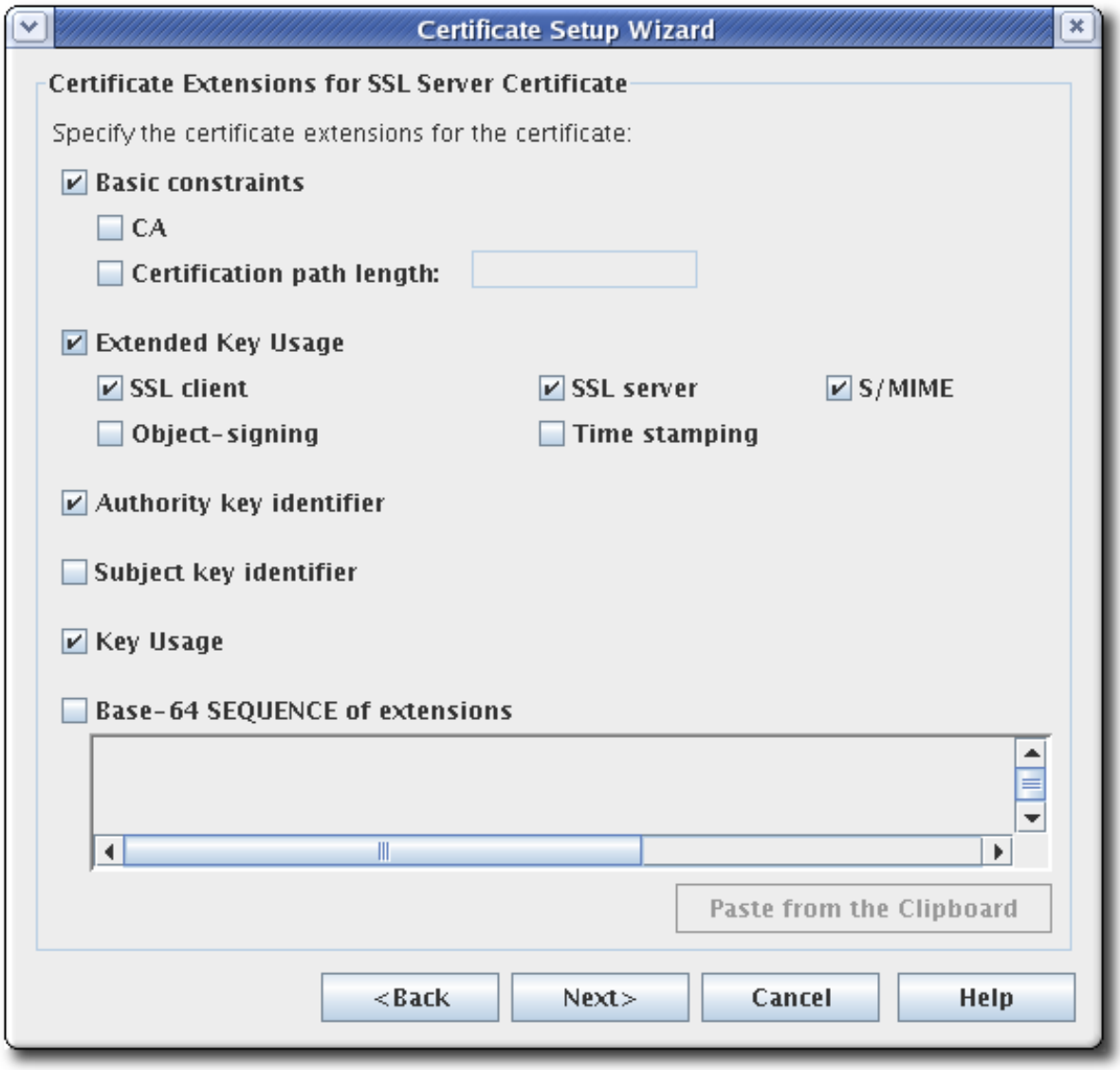
指定证书的有效期的开始和结束日期，以及有效期将在这些日期开始和结束的时间。



默认有效期为五年。

12.

为证书设置标准扩展。默认会选择所需的扩展。要更改默认选择，请参阅 附录 B, 证书和 CRL 的默认值、约束和扩展 中所述的指南。



注意

设置 CA 层次结构需要证书扩展。子 CA 必须具有证书，其中包括将其识别为从属 SSL CA （允许它们为 SSL CA 发布证书）或从属电子邮件 CA （允许它们为安全电子邮件 CA 发布证书的扩展）。禁用证书扩展意味着无法设置 CA 层次结构。

- *基本限制。相关的字段是 CA 设置以及认证路径长度的数字设置。*
- *扩展的密钥用法.*
- *颁发机构密钥标识符。*

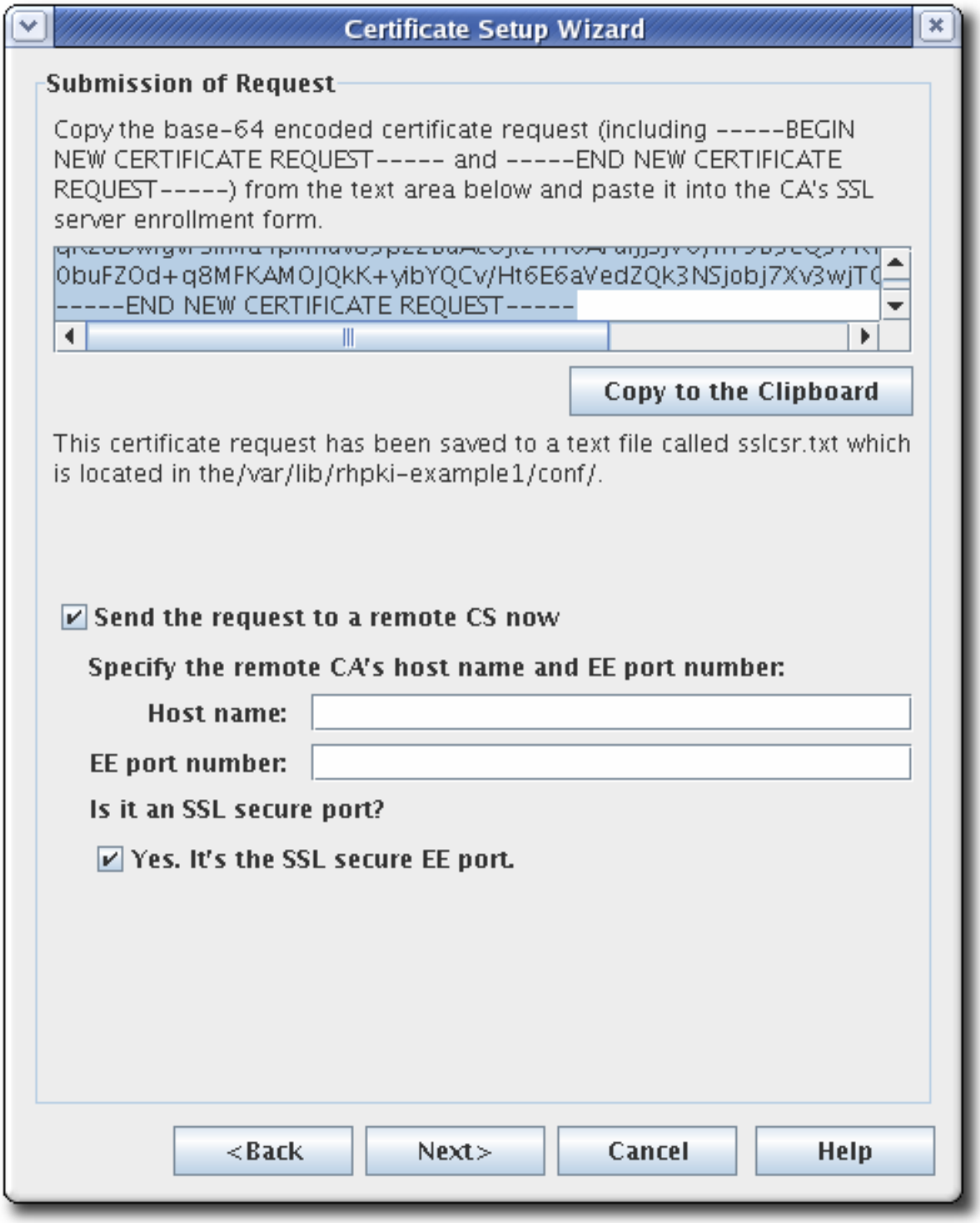
- *主题密钥标识符。*

- *密钥用法.数字签名（以 0 位）、非代表（位 1）、密钥证书签名（以 5 位）和 CRL 符号（以 6 位）位被默认设置。该扩展被标记为 PKIX 标准和 RFC 2459 建议的关键。有关 Key Usage 扩展的描述，请参阅 RFC 2459。*

- *扩展的 base-64 SEQUENCE。这适用于自定义扩展。将 MIME 64 DER 编码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 PKCSHQ10 格式，由标记行 -----BEGIN NEW CERTIFICATE REQUEST----- 和 ----- END NEW REQUEST----- REQUEST-----。例如：

> *-----BEGIN NEW CERTIFICATE REQUEST-----*
> *MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXR*
> *zY2FwZSBDb21tdW5pY2*
> *F0aW9uczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAw*
> *MFoXDTk5MDIyMzE5MDA*
> *wMnbjdgngYoxIDAeBgNVBAoTF05IdHNjYXBIIENvbW11bmljYXRpb25zMQ8wDQYDVQ*
> *QLEwZQZW9wbGUxFz*
> *AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjM*

```
CEGCSqGSIb3Dbndg
JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4
QgEBAQHBAQDAgCAM
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hlBgdR+
jWeLmD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

该向导还会将证书请求复制到它在配置目录中创建的文本文件，它位于 /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 17.1 “为证书签名请求创建的文件” 中。

表 17.1. 为证书签名请求创建的文件

filename	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。可以通过向导自动提交请求，或复制到剪贴板，并通过其终端实体页面手动提交到 CA。

注意

向导的 auto-submission 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。要将它提交到第三方 CA，请使用证书请求文件。

14. 检索证书。

    a. 打开 证书管理器结束实体页面。

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

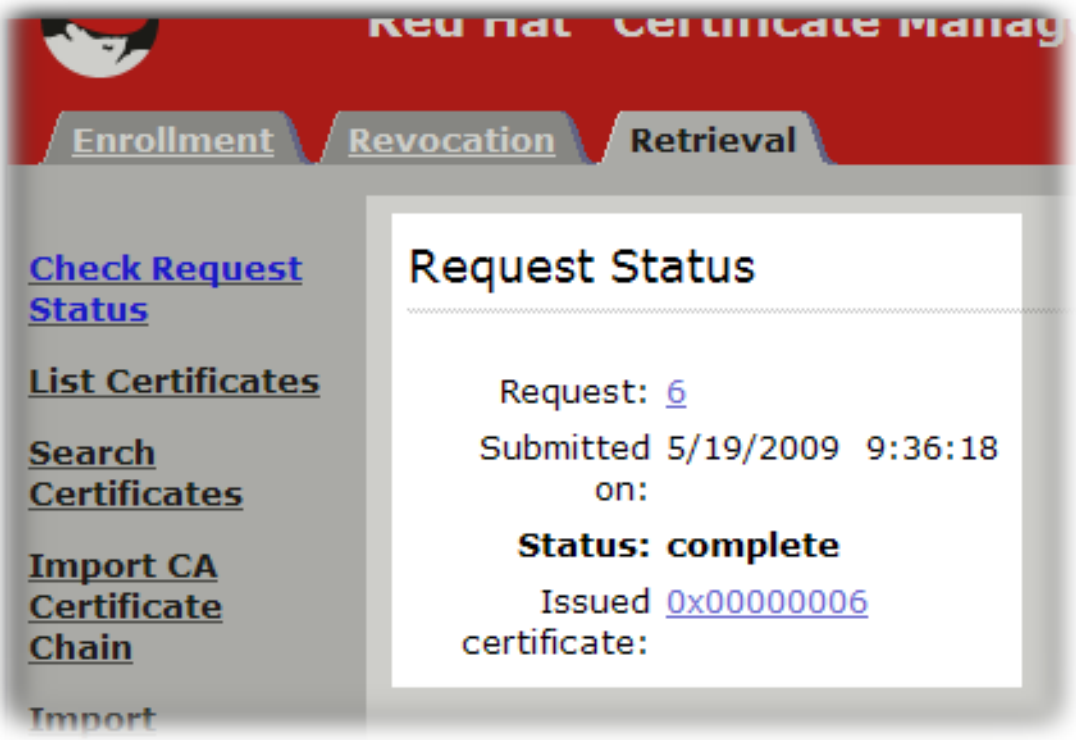
    b. 点 Retrieval 选项卡。

c.

填写提交证书请求时创建的请求 ID 号，然后单击 **Submit**。

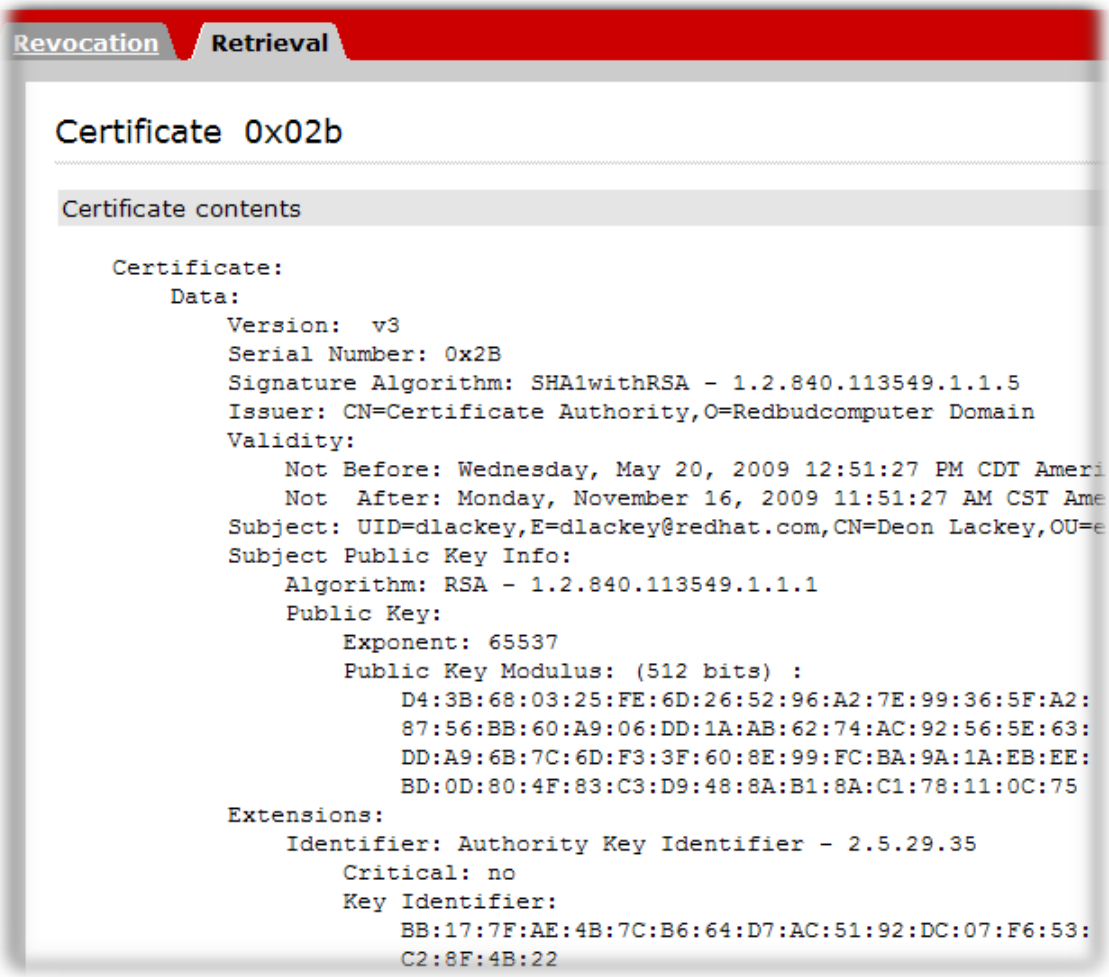
d.

下一页显示证书请求的状态。如果状态 完成，则有指向证书的链接。点 **Issued certificate** 链接。



e.

新证书信息以用户为**print** 格式显示，采用 **base-64** 编码格式，并且以 **PKCS the7** 格式显示。



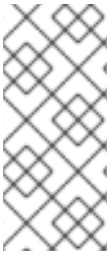
f.

将 base-64 编码证书（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子系统的内部数据库中。请参阅 第 15.3.2.1 节 “创建用户”。

注意

pkiconsole 已被弃用。

17.2.2. 请求其他证书



注意

用户从稍后使用的计算机生成并提交客户端请求非常重要，因为请求进程的一部分在本地计算机上生成私钥。如果需要位置 independence，请使用硬件令牌（如智能卡）来存储密钥对和证书。

1.

打开子系统控制台。例如：

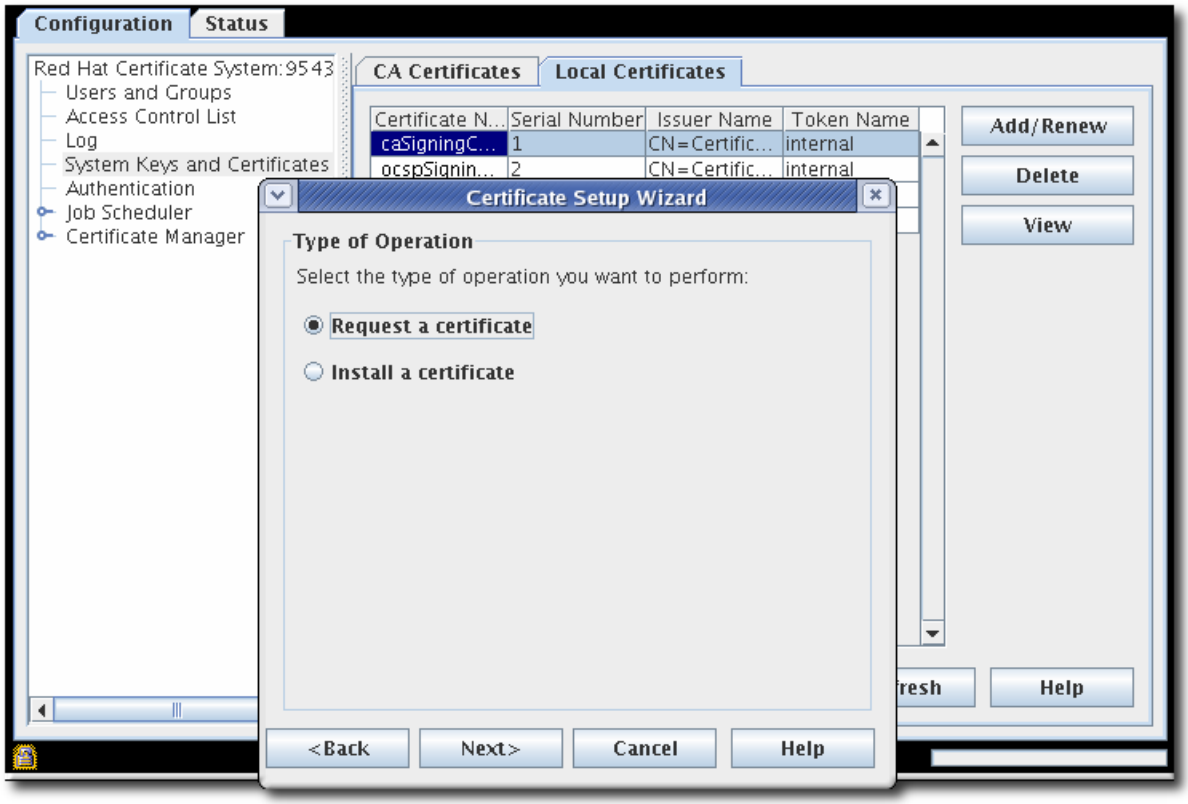
-

```
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。

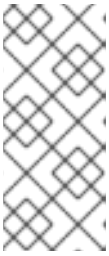
3. 在右侧面板中，选择 Local Certificates 选项卡。

4. 单击 Add/Renew。



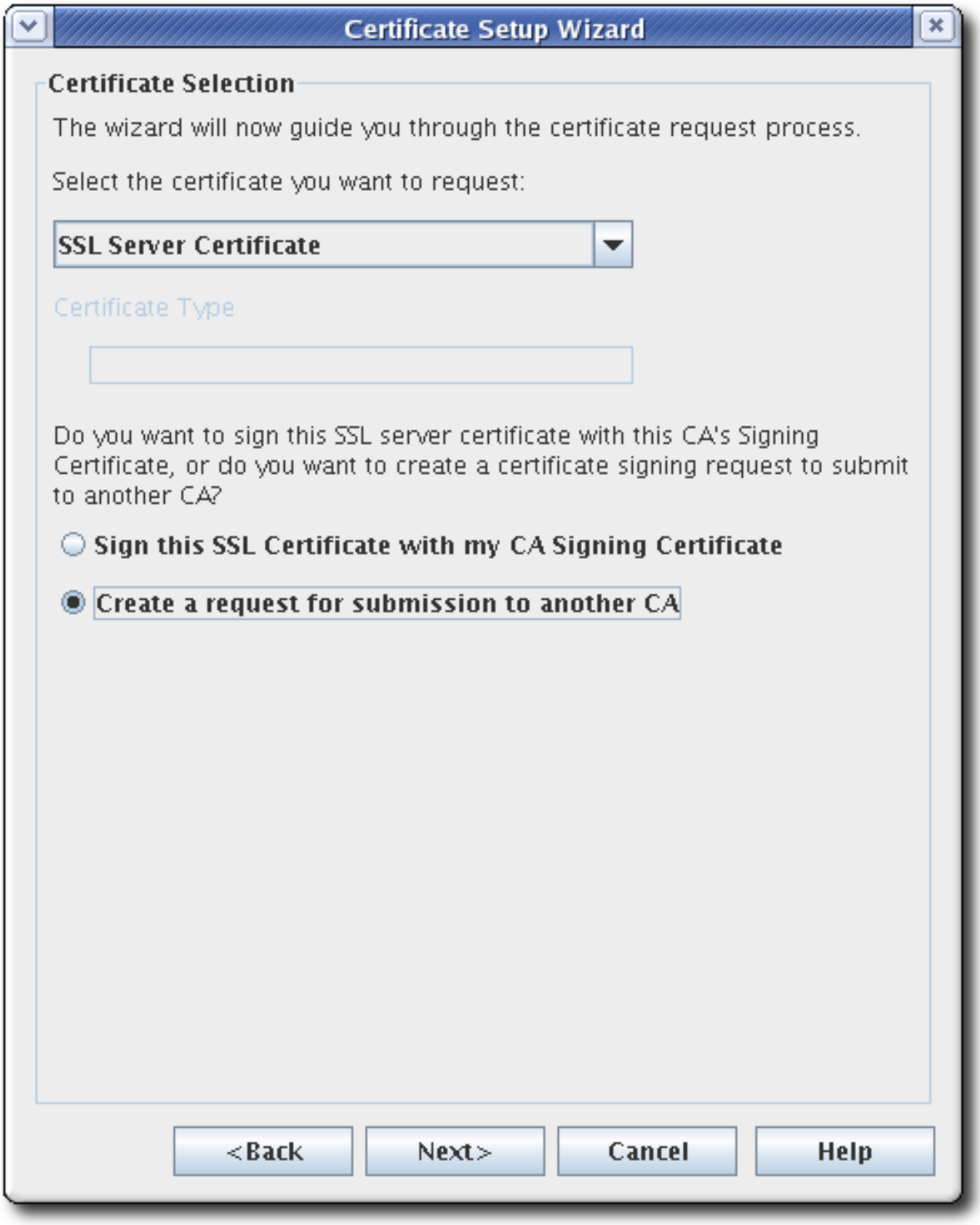
5. 选择 Request a certificate 单选按钮。

6. 选择要请求的证书类型。可以请求的证书类型因子系统而异。



注意

如果选择创建"其他"证书，则证书 类型字段 将变为活动状态。填写要创建的证书类型，可以是 CRL 签名证书的 caCrlSigning、caSignedLogCert （审计日志签名证书）或 SSL 客户端证书的客户端。



7.

选择哪个类型的 CA 将为请求签名。这些选项是使用本地 CA 签名证书，或者创建请求来提交到另一个 CA。

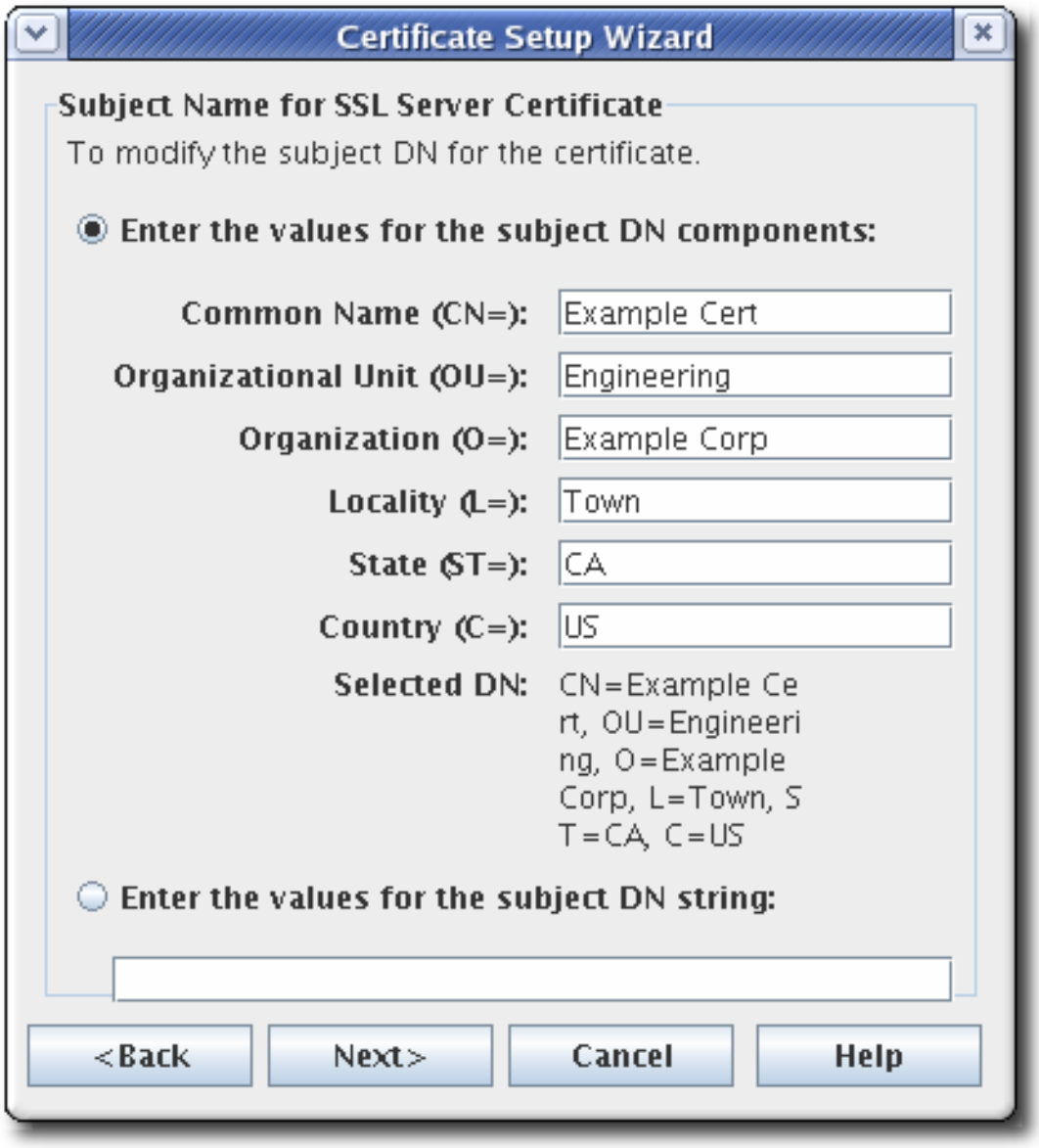
8.

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，也可以是列出的外部令牌之一。

若要创建新证书，您必须创建一个新密钥对。使用现有密钥对将只续订现有证书。

9.

指定主题名称。为单个 DN 属性输入值来构建主题 DN 或输入完整的字符串。



注意

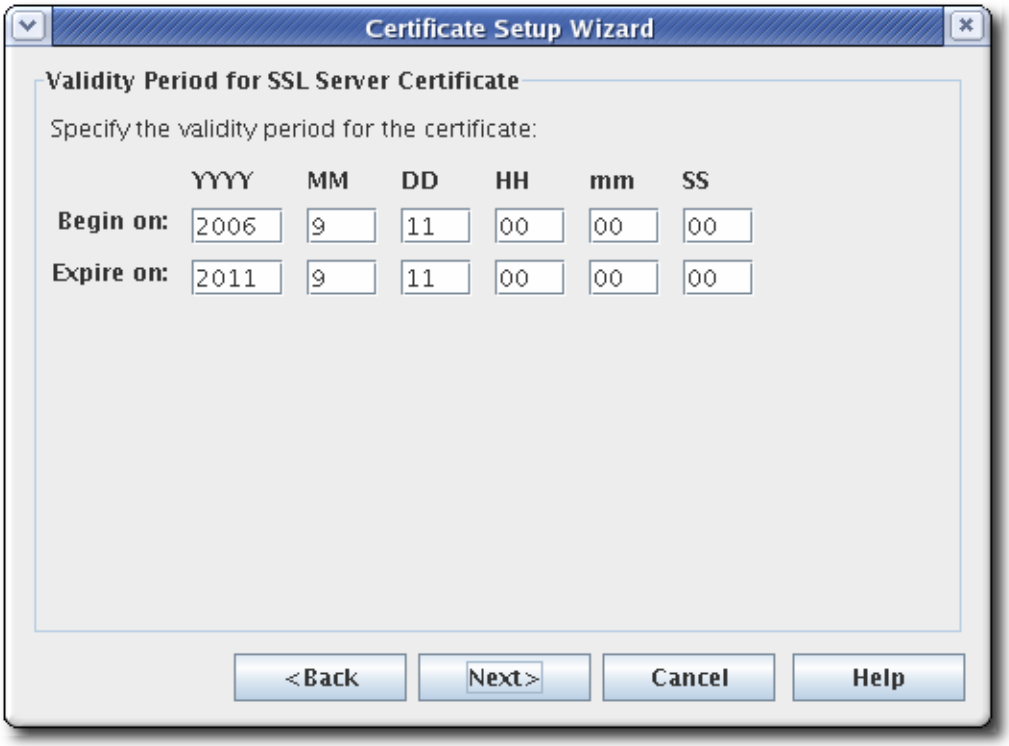
对于 SSL 服务器证书，通用名称必须是证书系统的完全限定域名，格式为 machine_name.domain.domain。

CA 证书请求表单支持通用名称、机构单元和请求名称字段的所有 UTF-8 字符。

这个支持不包括支持国际化域名。

10.

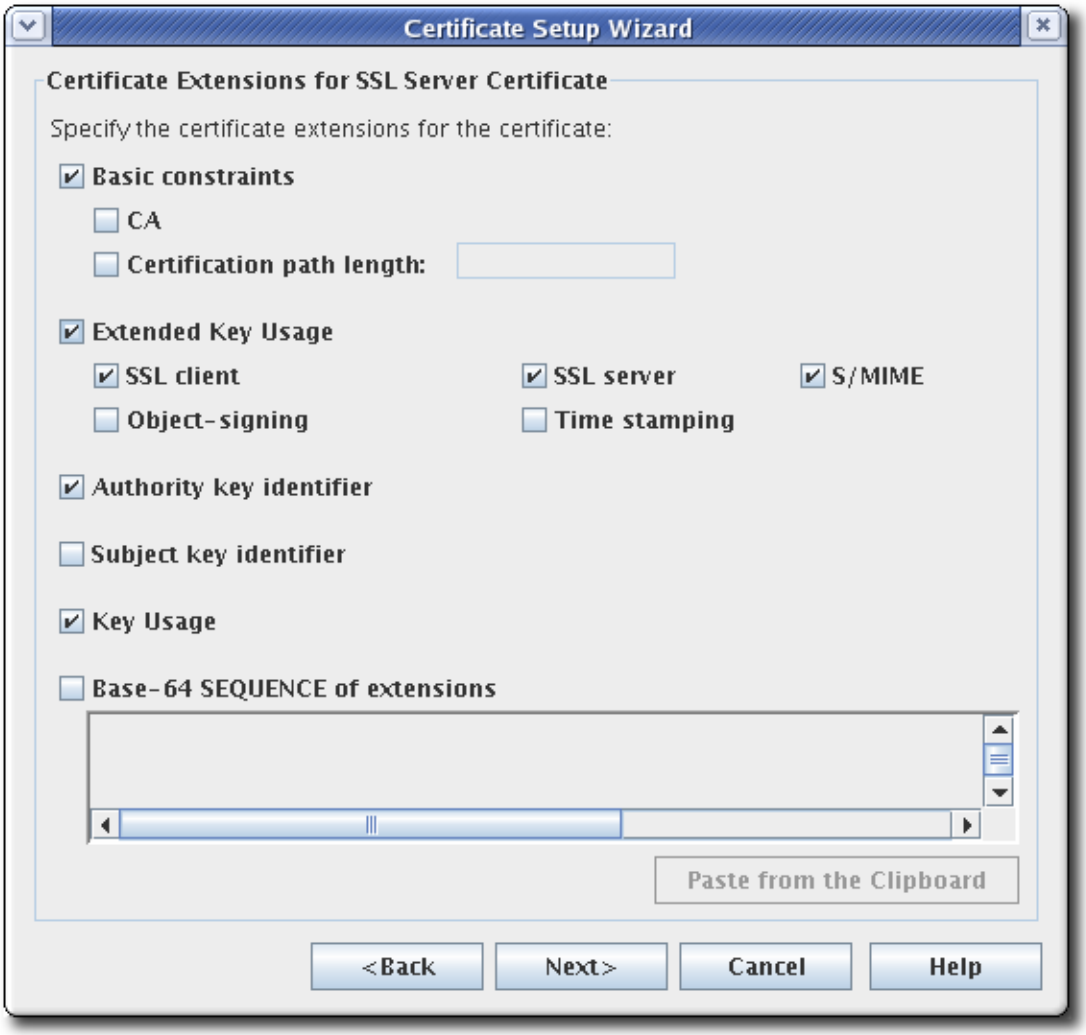
指定证书的有效期的开始和结束日期，以及有效期将在这些日期开始和结束的时间。



默认有效期为五年。

11.

为证书设置标准扩展。默认会选择所需的扩展。要更改默认选择，请参阅 附录 B, 证书和 CRL 的默认值、约束和扩展 中所述的指南。



- *扩展的密钥用法.*

- *颁发机构密钥标识符。*

- *主题密钥标识符。*

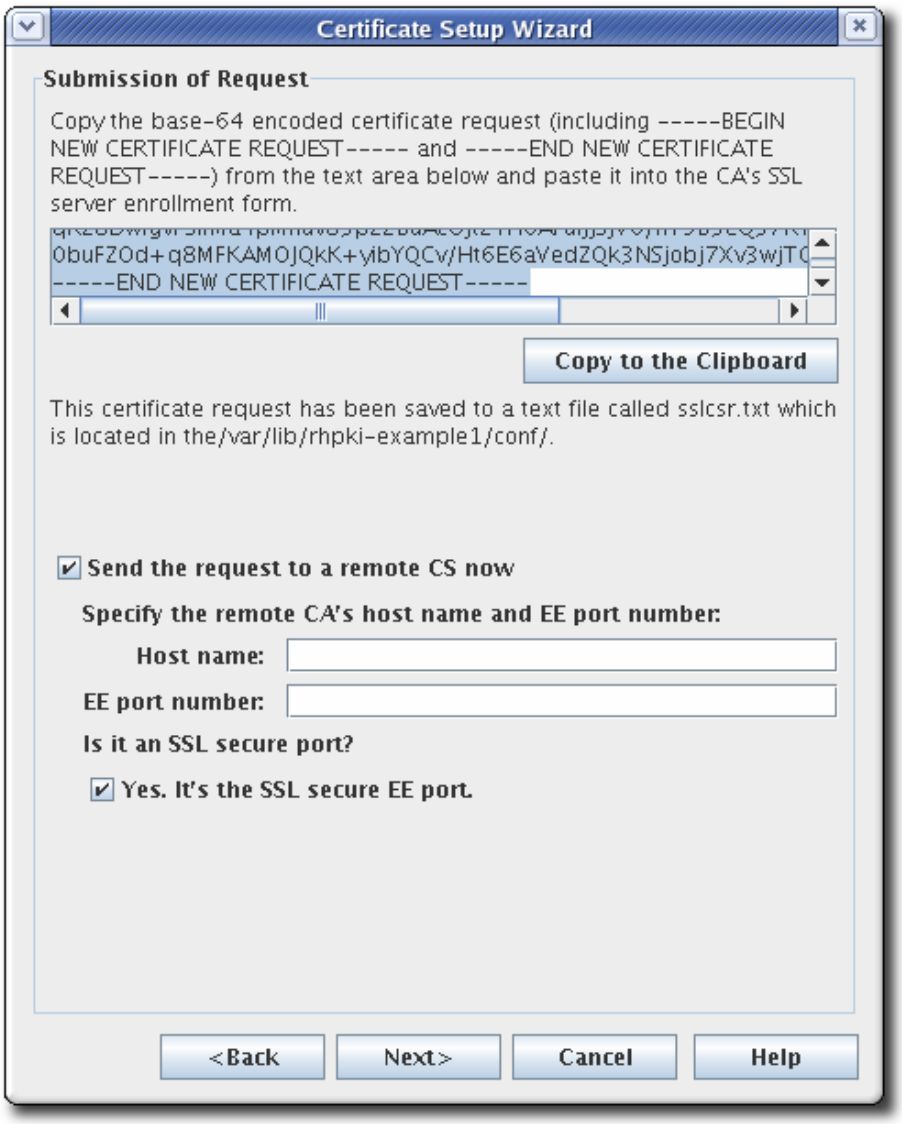
- *密钥用法.数字签名（以 0 位）、非代表（位 1）、密钥证书签名（以 5 位）和 CRL 符号（以 6 位）位被默认设置。该扩展被标记为 PKIX 标准和 RFC 2459 建议的关键。有关 Key Usage 扩展的描述，请参阅 RFC 2459。*

- *扩展的 base-64 SEQUENCE。这适用于自定义扩展。将 MIME 64 DER 编码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 PKCSHQ10 格式，由标记行 -----BEGIN NEW CERTIFICATE REQUEST----- 和 ----- END NEW REQUEST----- REQUEST-----。例如：

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXRzY2F
wZSBDb21tdW5pY2
F0aW9uczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAwMFo
XDTk5MDIyMzE5MDA
wMnbjdgngYoxIDAeBgNVBAoTF05ldHNjYXBlIENvbW11bmljYXRpb25zMQ8wDQYDVQQLEw
ZQZW9wbGUxFz
AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjMCEGC
SqGSIb3Dbndg
JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4QgEB
AQHBAQDAgCAM
```

```
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hIBgdR+jWeL
mD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

该向导还会将证书请求复制到它在配置目录中创建的文本文件，它位于 /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 17.2 “为证书签名请求创建的文件” 中。

表 17.2. 为证书签名请求创建的文件

filename	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。可以通过向导自动提交请求，或复制到剪贴板，并通过其终端实体页面手动提交到 CA。

注意

向导的 auto-submission 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。要将请求提交到第三方 CA，请使用其中一个证书请求文件。

13. 检索证书。

a. 打开 证书管理器结束实体页面。

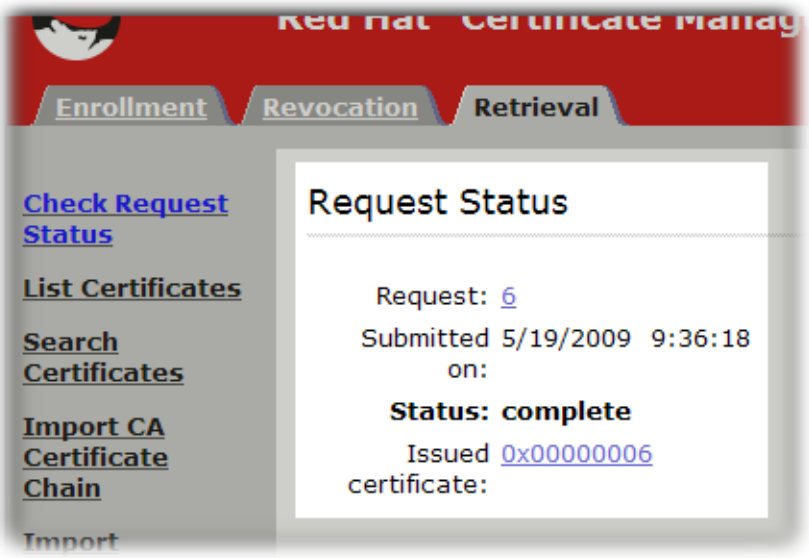
```
https://server.example.com:8443/ca/ee/ca
```

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。

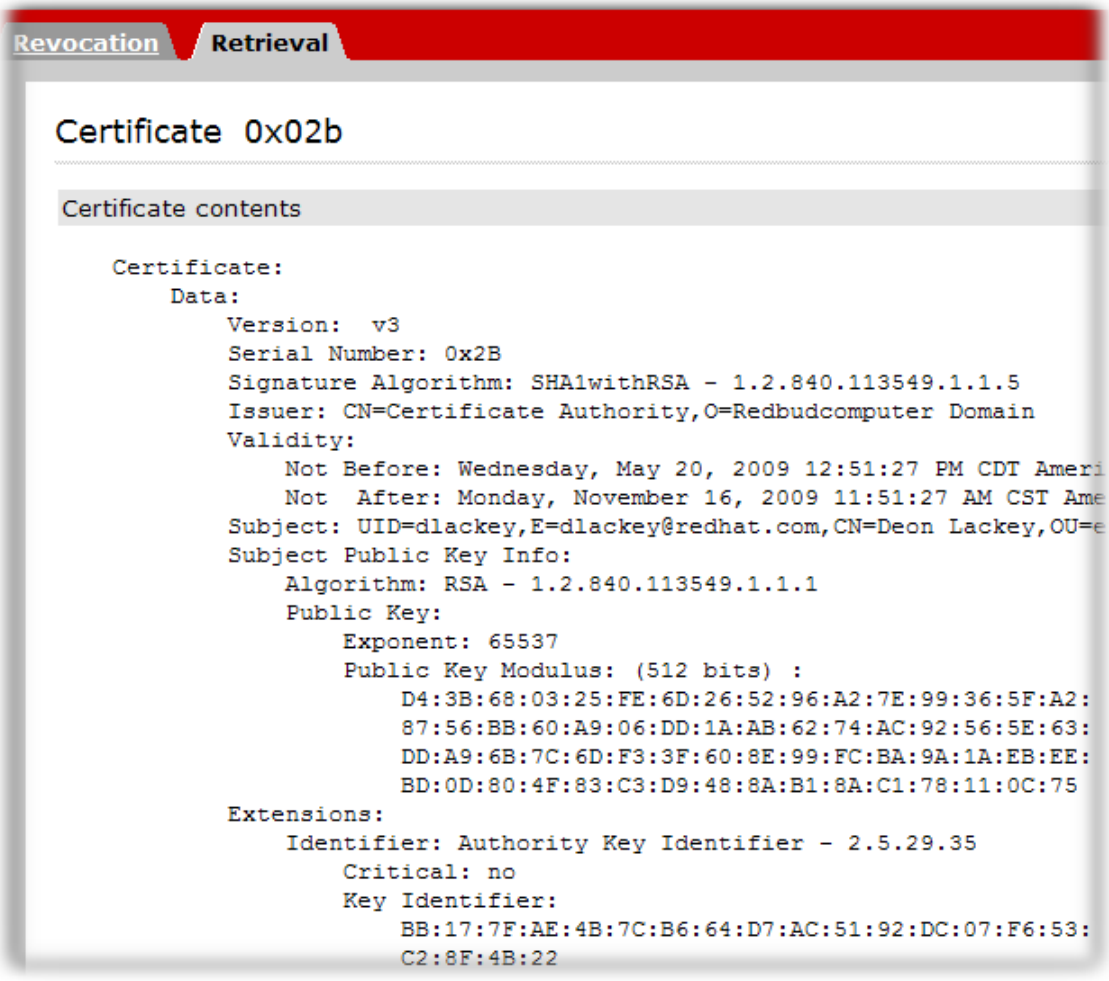
d.

下一页显示证书请求的状态。如果状态 完成，则有指向证书的链接。点 Issued certificate 链接。



e.

新证书信息以用户为print 格式显示，采用 base-64 编码格式，并且以 PKCS the7 格式显示。



f.

将 base-64 编码证书（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子系统的内部数据库中。请参阅 第 15.3.2.1 节 “创建用户”。

17.3. 续订子系统证书

更新证书的方法有两种。重新生成证书 会取其原始密钥及其原始配置集和请求，并使用新的有效期和过期日期重新创建相同的密钥。重新加密 证书将初始证书请求重新提交到原始配置文件，但会生成一个新密钥对。管理员可以通过重新密钥来更新管理员证书。

17.3.1. 在 End-Entities 表单中重新加密证书

可以使用原始证书的序列号直接在最终用户注册表单中直接续订子系统证书。

1.

以 CA 结束实体形式续订证书，如 第 5.4 节 “续订证书” 所述。这要求正在续订的子系统证书的序列号。

2.

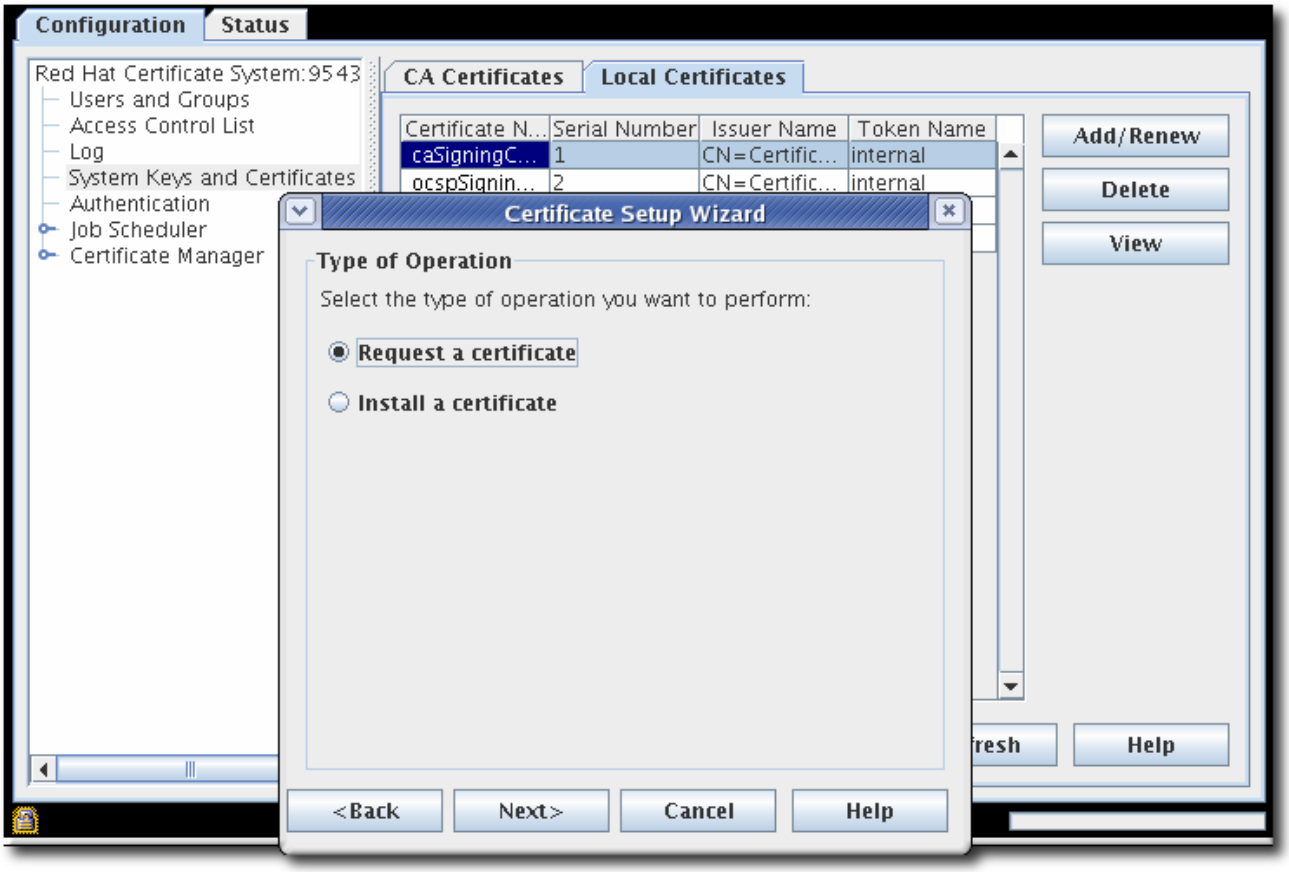
将证书导入到子系统的数据库，如 第 17.6.1 节 “在证书系统数据库中安装证书” 所述。证书可以使用 certutil 或控制台导入。例如：

```
certutil -A -n "ServerCert cert-example" -t u,u,u -d /var/lib/pki/instance_name/alias -a -i
/tmp/example.cert
```

17.3.2. 在控制台中续订证书

Java 子系统可以通过其管理控制台更新其任何子系统证书。这个过程与请求新的子系统证书 (第 17.2 节 “通过控制台请求证书”)完全相同，有一个关键区别：续订 使用现有密钥对 而不是生成新的密钥对。

图 17.1. 续订子系统证书



续订证书后，从数据库中删除原始证书(第 17.6.3 节 “从数据库中删除证书”)。

17.3.3. 使用 certutil 续订证书

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，可以通过常规配置文件页面提交新证书请求，以便 CA 发布更新的证书。

注意

加密和签名证书在单个步骤中创建。但是，续订过程一次仅续订一个证书。

要在证书对中续订这两个证书，必须单独更新每个证书。

1. *获取令牌数据库的密码。*

```
cat /var/lib/pki/instance_name/conf/password.conf

internal=263163888660
```

2. 打开证书正在续订的实例的证书数据库目录。

```
cd /var/lib/pki/instance_name/alias
```

3. 列出正在续订的证书的密钥和别名。要续订证书，用于生成和提供给新证书的主题名称必须与旧证书中的主题名称相同。

```
# 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和续订证书之间的区别在于 **-n** 选项的值。要生成完全新的请求和密钥对，**-k** 设置密钥类型，与 **-g** 一起使用，这会设置位长度。对于续订请求，**-n** 选项使用证书别名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 5.3 节 “请求和接收证书” 所述。

17.3.4. 续订系统证书

在 PKI 服务器运行时，证书系统不会自动在线续订系统证书。但是，如果系统证书过期，证书系统将无法启动。

续订系统证书：

1. 如果系统证书已过期：

   a. 创建临时证书：

   ```
   # pki-server cert-create sslserver --temp
   ```

   b. 将临时证书导入到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：

   ```
   # pki-server start instance_name
   ```

2. 显示证书并记录过期系统证书的 ID：

   ```
   # pki-server cert-find
   ```

3. 创建新的持久性证书：

   ```
   # pki-server cert-create certificate_ID
   ```

4. 停止证书系统：

   ```
   # pki-server stop instance_name
   ```

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：

```
# pki-server start instance_name
```

17.4. 更改子系统证书的名称

续订证书的一种替代方案是使用新证书替换它们，这意味着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中，并且旧的证书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 nickname 保持不变，服务器也可以找到所需的证书，即使其他因素（如主题名称、序列号或密钥 - ）不同。

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 配置文件中的所有所需设置中更新证书 nickname。

重要

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 17.3 “CA Certificate Nickname 参数”
- 表 17.4 “KRA Certificate Nickname 参数”
- 表 17.5 “OCSP Certificate Nickname 参数”
- 表 17.6 “TKS 证书 Nickname 参数”
- 表 17.7 “CS.cfg 中的 TPS Nickname 参数”

表 17.3. CA Certificate Nickname 参数

CA 签名证书	• ca.cert.signing.nickname • ca.signing.cacertnickname • ca.signing.certnickname • ca.signing.nickname • cloning.signing.nickname
OCSP 签名证书	• ca.ocsp_signing.cacertnickname • ca.ocsp_signing.certnickname • ca.cert.ocsp_signing.nickname • ca.ocsp_signing.nickname • cloning.ocsp_signing.nickname
子系统证书	• 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
审计签名证书	• ca.audit_signing.nickname • ca.cert.audit_signing.nickname • cloning.audit_signing.nickname

表 17.4. 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，则 TKS 配置中需要更改，即使 TKS 证书都保持不变。
存储证书	• cloning.storage.nickname • kra.storage.nickname • kra.cert.storage.nickname
服务器证书	• kra.cert.sslserver.nickname • kra.sslserver.nickname
子系统证书	• 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 17.5. OCSP Certificate Nickname 参数

OCSP 签名证书	• cloning.signing.nickname • ocsp.signing.certnickname • ocsp.signing.cacertnickname • ocsp.signing.nickname

服务器证书	• ocsp.cert.sslserver.nickname • ocsp.sslserver.nickname
子系统证书	• 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 17.6. 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
子系统证书	• 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，则 TKS 配置中需要更改，即使 TKS 证书都保持不变。	

表 17.7. CS.cfg 中的 TPS Nickname 参数

服务器证书	• tps.cert.sslserver.nickname
子系统证书	• 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

17.5. 使用跨证书

在未来 1990s 中，因为美国政府开始增强其公钥基础架构，它很明显地利用他们自己的政府分支，单独的 PKI 部署仍然需要象从其自己的 CA 发布证书一样识别并信任其他证书。（获取网络外部使用的证书的方法是严重的，不能容易解决任何 PKI 管理员的问题。）

美国政府制定了一个标准，用于发布跨对证书，称为联邦信息处理标准证书颁发机构。出于明显的原因，这些证书也称为网桥证书。网桥或跨对证书的 CA 签名证书是帧为双证书对的 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。合作伙伴 CA 将其他 CA 签名证书存储在其数据库中，因此其他 PKI 中发布的所有证书都可以被信任并识别。

桥接证书遵循 CA 在其自己的 PKI 中未串联到 root CA 的 CA 发布的证书。通过跨对 CA 证书在证书系统 CA 和另一个 CA 之间建立信任，可以下载跨修复证书并用来信任由其他 CA 发布的证书，就像下载和安装 CA 发布的所有 CA 证书一样。

证书系统可以发布、导入和发布跨对 CA 证书。必须创建一个特殊的配置文件来发布跨对证书，然后使用 CA 子系统的证书向导为 CA 请求并安装证书。

有关创建跨对证书配置文件的更多信息，请参阅 Red Hat Certificate System Planning, Installation, and Deployment Guide 中的 Configuring cross-Pair profile 部分。

有关发布跨对证书的更多信息，请参阅 第 9.9 节 “发布跨证书”。

17.5.1. 安装跨证书

可以使用 certutil 工具或从证书设置向导中选择 跨证书设置向导中的跨证书 数据库导入到证书系统数据库中，如 第 17.6.1 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时，一个 crossCertificatePair 条目会被形成并存储在数据库中。创建 crossCertificatePair 条目后，原始单个跨对 CA 证书会被删除。

17.5.2. 搜索跨证书

网桥证书中的两个 CA 都可以作为 LDAP 数据库中的 跨CertificatePair 条目存储或发布跨对证书。证书管理器的内部数据库可以使用 ldapsearch 搜索 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=*)"
```

17.6. 管理证书数据库

每个证书系统实例都有一个证书数据库，在其内部令牌中维护。此数据库包含属于证书系统实例中安装的子系统的证书，以及子系统用来验证它们接收的证书的各种 CA 证书。

即使外部令牌用于生成和存储密钥对，证书系统也始终在其内部令牌中维护其可信和不受信任的 CA 证书的列表。

本节介绍如何查看证书数据库的内容，删除不需要的证书，以及使用证书系统窗口更改数据库中安装的 CA 证书的信任设置。有关向数据库添加证书的详情，请参考 第 17.6.1 节 “在证书系统数据库中安装证书”。

注意

证书系统命令行工具 certutil 可以用来通过编辑信任设置和添加和删除证书来管理证书数据库。有关此工具的详情，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，以确保它不包含任何不需要的 CA 证书。例如，如果数据库包含应在 PKI 设置中不被信任的 CA 证书，请删除它们。

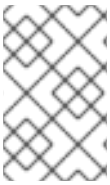
17.6.1. 在证书系统数据库中安装证书

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 工具在子系统证书数据库中安装。

- 第 17.6.1.1 节 “通过控制台安装证书”
- 第 17.6.1.2 节 “使用 certutil 安装证书”
- 第 17.6.1.3 节 “关于 CA 证书链”

17.6.1.1. 通过控制台安装证书



注意

pkiconsole 已被弃用。

证书设置向导可安装或将以下证书导入到证书系统实例使用的内部或外部令牌中：

- 证书系统子系统使用的任何证书
- 来自外部 CA 或其他证书系统 CA 的任何可信 CA 证书
- 证书链

证书链包含一组证书：主题证书、可信 root CA 证书以及将主题证书链接到可信 root 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；任何证书都不能是用户证书。

在证书链中，链中的每个证书编码为单独的 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，所有方法都导入到最后一个证书，也可能不是 root CA 证书。如果链中的任何证书已安装在本地证书数据库中，向导会将现有证书替换为链中的证书。如果链包含中间 CA 证书，向导会将它们作为 不受信任的 CA 证书添加到证书数据库中。

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. 有两个选项卡可以安装证书，具体取决于子系统类型和证书的类型。

   - CA 证书 选项卡用于安装 CA 证书和密钥链。对于证书管理器，此选项卡用于第三方 CA 证书或其他证书系统 CA 证书；所有本地 CA 证书都安装在本地 证书 选项卡中。对于所有其他子系统，所有 CA 证书和密钥都通过此选项卡安装。

   - Local Certificates 选项卡安装了所有服务器证书、子系统证书和本地证书，如 OCSP 签名或 KRA 传输。

   选择适当的选项卡。

4. 要在 Local Certificates 选项卡中安装证书，请单击 Add/Renew。要在 CA Certificates 选项卡中安装证书，请单击 Add。两者都将打开证书设置向导。

   a. 当向导打开时，选择 Install a certificate 单选按钮，然后点 Next。

   b. 选择要安装的证书类型。下拉菜单的选项与创建证书的选项相同，具体取决于子系统类型，以及安装跨对证书的附加选项。

   c. 粘贴到证书正文中，包括 -----BEGIN CERTIFICATE----- 和 -----END CERTIFICATE-----、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
NVBAYTAIVTMREwDwYDVQQKEwhOZXRzY2FwZTENMAsGA1UECxMEUHawcz
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.

任何签署证书的 CA 都必须被子系统信任。确保此 CA 的证书数据库（内部或外部）中存在，并且它被信任。

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 CA 的证书但不被信任，请将信任设置改为 trusted，如 第 17.7 节 “更改 CA 证书的信任设置” 所示。

当安装不是由证书系统证书数据库中存储的 CA 发布的证书时，请将该 CA 的证书链添加到数据库中。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。

17.6.1.2. 使用 certutil 安装证书

要使用 certutil 在证书系统实例的安全数据库中安装子系统证书，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，-i 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

如果证书系统实例的证书和密钥存储在 HSM 上，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。

17.6.1.3. 关于 CA 证书链

支持证书的任何客户端或服务器软件在其证书数据库中维护一组可信 CA 证书。这些 CA 证书决定了软件可以验证的其他证书。在最简单的情形中，软件只能验证其具有证书的 CA 发布的证书。可信 CA 证书也可以是 CA 证书链的一部分，每个证书都使用证书层次结构中的 CA 发布。

链中的第一个证书以特定于上下文的方式进行处理，这因导入的方式而异。对于 Mozilla Firefox，这种处理取决于所下载对象的 MIME 内容类型。对于红帽服务器，它取决于服务器管理界面中选择的选项。

后续证书都被视为相同。如果证书在 Netscape 证书类型证书扩展中包含 SSL-CA 位，且本地证书数据库中尚不存在，则它们会被添加为不受信任的 CA。只要链中存在信任的 CA，它们就可以进行证书链验证。

17.6.2. 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 cert9.db 的证书。或者，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 17.6.2.1 节 “通过控制台查看数据库内容”

- 第 17.6.2.2 节 “使用 certutil 查看数据库内容”

注意

cert9.db 数据库中列出的证书是用于子系统操作的子系统证书。客户端证书与 LDAP 内部数据库中的用户条目一起存储。

17.6.2.1. 通过控制台查看数据库内容

注意

pkiconsole 已被弃用。

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
```

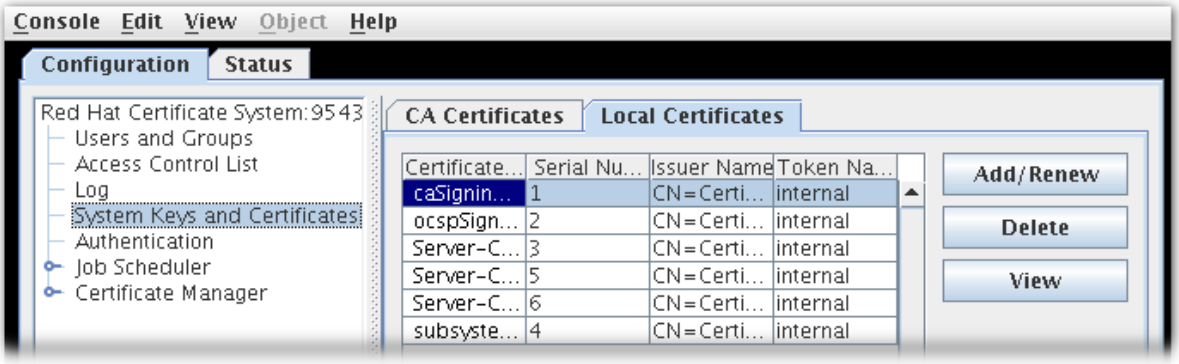
2. 在 Configuration 选项卡中，从左侧导航树中选择 System Keys and Certificates。

3. CA 证书和本地证书 有两种选项卡，它列出了不同类型的证书。

   - CA 证书列出了对应的私钥材料不可用的 CA 证书，如第三方 CA 发布的证书，如 Entrust 或 Verisign 或外部证书系统证书管理器。

   - 本地 证书列出了证书系统子系统实例保存的证书，如 KRA 传输证书或 OCSP 签名证书。

图 17.2. 证书验证 Tab



4. 证书数据库管理 表列出了子系统上安装的所有证书。每个证书都会提供以下信息：

- 证书名称
- 序列号
- 签发者名称，这是此证书签发者的通用名称(cn)。
- 令牌名称，包含证书的加密令牌的名称；对于数据库中存储的证书，这是 内部的。

要查看有关证书的更多详细信息，请选择证书，然后单击 View。这会打开一个窗口，显示证书的序列号、有效期、主题名称、签发者名称和证书指纹。

17.6.2.2. 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，请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。

17.6.3. 从数据库中删除证书

删除不需要的证书可减少证书数据库的大小。

注意

从证书数据库中删除 CA 证书时，要小心不要删除 中间 CA 证书，这有助于将子系统链到可信 CA 证书。如果有疑问，请将数据库中的证书保留为 不受信任的 CA 证书；请参阅 第 17.7 节 “更改 CA 证书的信任设置”。

- 第 17.6.3.1 节 “通过控制台删除证书”
- 第 17.6.3.2 节 “使用 certutil 删除证书”

17.6.3.1. 通过控制台删除证书

注意

pkiconsole 已被弃用。

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，然后单击 Delete。

4. 出现提示时，确认删除。

17.6.3.2. 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. 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。

17.7. 更改 CA 证书的信任设置

证书系统子系统使用其证书数据库中的 CA 证书来验证在启用了 SSL 的通信期间收到的证书。

可能需要临时或永久更改证书数据库中存储的 CA 上的信任设置。例如，如果访问或被破坏的证书存在问题，请将 CA 证书标记为不受信任的，可防止使用这个 CA 签名的证书向证书系统进行身份验证的实体。当问题被解决时，CA 可以再次标记为可信。

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 17.6.3 节 “从数据库中删除证书”。

17.7.1. 通过控制台更改信任设置

注意

pkiconsole 已被弃用。

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，从左侧导航树中，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，然后单击 Edit。

5. 提示会打开哪个读取 证书链(un)信任是否是(un) trust？

   点 yes 更改证书链的信任设置；按 不 保留原始信任关系。

17.7.2. 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. 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。

17.8. 管理子系统使用的令牌

证书系统管理器两组令牌：子系统使用的令牌来执行通过子系统发布的 PKI 任务和令牌。这些管理任务专门引用子系统使用的令牌。

有关管理智能卡令牌的详情，请参考 第 6 章 使用和配置令牌管理系统：TPS 和 TKS。

17.8.1. 检测令牌

要查看证书系统是否可以检测到令牌，请使用 TokenInfo 工具。

```
TokenInfo /var/lib/pki/instance_name/alias
Database Path: /var/lib/pki/instance_name/alias
Found external module 'NSS Internal PKCS #11 Module'
```

此工具将返回证书系统可检测到的所有令牌，而不仅仅是在证书系统中安装的令牌。

17.8.2. 查看令牌

要查看当前为证书系统实例安装的令牌列表，请使用 modutil 实用程序。

1.
打开实例 别名 目录。例如：

```
cd /var/lib/pki/instance_name/alias
```

2.
使用 modutil 工具显示安装的 PKCS the 模块的信息，以及使用 modutil 工具有关对应令

牌的信息。

```
modutil -dbdir . -nocertdb -list
```

17.8.3. 更改令牌的密码

存储子系统的密钥和证书的令牌（内部或外部）受密码保护（加密）。要解密密钥对或获得对它们的访问权限，请输入令牌密码。当令牌首次访问时（通常在证书系统安装过程中）时设定此密码。

最好更改用于定期保护服务器的密钥和证书的密码。更改密码可最大程度降低发现密码的人员的风险。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/。

单点登录密码缓存将令牌密码存储在 password.conf 文件中。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 Red Hat Certificate System Planning、安装和部署指南。

第 18 章 在 RED HAT ENTERPRISE LINUX 7 中设置时间和日期

这部分包含如何在 Red Hat Enterprise Linux 7 中设置时间和日期：

系统时间始终保留在 协调通用时间 (UTC)中，并根据需要将应用程序转换为本地时间。本地时间 是当前时区的实际时间，考虑日常 节省时间 (DST)。

timedatectl 工具作为 systemd 系统和服务管理器的一部分发布，允许您检查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，将 SS 替换为一秒钟，以两位形式输入。

更改当前日期

```
timedatectl set-time YYYY-MM-DD
```

使用四位数字替换 YYYY，使用两位月替换 MM，而使用月的两位天替换 DD。

时间更改由操作系统审计。如需更多信息，请参阅 Red Hat Certificate System 规划、安装和部署指南中的 审计时间 更改事件部分。

第 19 章 确定证书系统产品版本

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 10.0 (Batch Update 1)
```

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，因此可以有单独的版本号。以上将显示每个当前运行的组件的版本号。

第 20 章 更新 RED HAT CERTIFICATE SYSTEM

要更新证书系统及其在其中运行的操作系统，请使用 yum update 命令。这会下载、验证并安装证书系统以及操作系统软件包的更新。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 Red Hat Certificate System 规划、安装和部署指南中的更新证书系统软件包部分。

第 21 章 故障排除

本章介绍了在安装证书系统时遇到的一些一些常见使用问题。

问：

初始化脚本返回一个 OK 状态，但我的 CA 实例没有响应。这是因为什么？

答：

这应该不会发生这种情况。通常（但不始终），这表示 CA 的监听程序问题，但可能会有很多不同的原因。检查实例的 catalina.out、系统 和调试 日志文件，以查看发生的错误。这列出了几个常见的错误。

一个情况是，当 CA 有一个 PID 时，表示进程正在运行，但没有为服务器打开任何监听程序。这将返回 catalina.out 文件中的 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的 JSS 版本或 NSS。进程在路径中需要 libnss3.so。使用以下命令检查它：

```
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
pki-server restart instance_name
```

问：

我无法打开 pkiconsole，在 stdout 中看到 Java 例外。

答：

这可能意味着您安装了错误的 JRE，或者将错误的 JRE 设置为默认值。运行 alternatives --config java 以查看所选的 JRE。Red Hat Certificate System 需要 OpenJDK 1.8。

问：

我试图运行 pkiconsole，并在 stdout 中得到套接字例外。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（例如，server.xml中存在错误的配置），或者给出了错误的端口来访问管理界面。

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：

我尝试注册证书，我收到错误 "request is not commit...Subject Name Not Found"?

答：

这通常在自定义 LDAP 目录身份验证配置集中发生，显示目录操作失败。特别是，它失败，因为它无法构建正常工作的 DN。这个错误将在 CA 的调试日志 中。例如，此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 - 主题 DN 中使用的这些组件都可能会导致失败。在大多数情况下，RHC 2253 中定义的 X.509 属性应该在主题 DN 中使用，而不是使用自定义属性。

问：

为什么我的注册的证书没有被发布？

答：

这通常表示 CA 被错误配置。要查找错误的主要位置是 调试日志，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 选项卡中检查发布配置。在本例中，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：

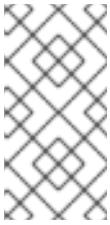
1. 设置 VNC 服务器，例如在 Red Hat Certificate System 服务器中。有关远程桌面访问的详情，请查看 RHEL 8 文档中的 相关部分。

重要

pkiconsole 工具无法在启用了联邦信息处理标准(FIPS)模式的服务器上运行。如果您的证书系统服务器上启用了 FIPS 模式，请使用 Red Hat Enterprise Linux 的不同主机来运行 VNC 服务器。请注意，这个工具将被弃用。

2. 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，红帽只支持从集成的软件仓库在 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，则会出现连接问题，或者发生 TLS 连接故障，则无法连接到依赖它的子系统实例。实例调试日志将特别识别 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（如电缆未插入）后，红帽目录服务器将停止、发生大量数据包丢失，或确保 TLS 连接可以被重新创建、停止，然后在问题中启动证书系统实例：

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

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

第 22 章 子系统控制和维护

本章介绍了如何控制（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。

22.1. 启动、停止、重启和获取状态

Red Hat Certificate System 子系统实例可以使用 Red Hat Enterprise Linux 8 上的 systemctl 工具停止并启动。

注意

您还可以使用 pki-server 别名启动和停止实例： pki-server <command> <instance> 是 systemctl & lt;command> pki-tomcatd@<instance>.service. 的别名。

启动实例：

```
# systemctl start unit_file@instance_name.service
```

```
# pki-server start instance_name
```

停止一个实例：

```
# systemctl stop unit_file@instance_name.service
```

```
# pki-server stop instance_name
```

重启实例：

```
# systemctl restart unit_file@instance_name.service
```

```
# pki-server restart instance_name
```

显示实例的状态：

```
# systemctl status unit_file@instance_name.service
```

unit_file 具有以下值之一：

- *pki-tomcat ：禁用 watchdog*
- *pki-tomcat-nuxwdog: 启用 watchdog*

22.2. 子系统健康检查

管理员定期监控可能的故障非常重要，如下所示：

- *由完整磁盘导致的审计失败*
- *由 HSM 连接问题导致的签名失败*
- *LDAP 服务器连接问题*
- *以此类推*

Self-tests 也可以根据需要运行，如 第 14.9 节 “运行自测试” 所述。

22.2.1. PKI 中的健康检查

PKI Healthcheck 是一个命令行工具，可帮助发现可能会影响证书系统环境健康状况的问题。如果需要，此工具可报告到 Red Hat Identity Management 中的 Healthcheck 工具。

22.2.1.1. PKI Healthcheck 测试模块

PKI Healthcheck 由独立模块组成，用于测试：

- *CS.cfg 和 NSS 数据库之间的证书同步*

检查 CS.cfg 中的系统证书（位于 /var/lib/pki/<instance>/<subsystem>/conf/CS.cfg）和 NSS 数据库（位于 /var/lib/pki/<instance>/alias/ ）匹配。否则，证书颁发机构(CA)无法启动。

- 系统证书过期

  检查安装的系统证书的到期状态（请参阅系统证书以了解更多信息）。

- 系统证书信任标志

  检查安装的系统证书是否使用正确的信任标志（请参阅系统证书以了解更多信息）。

- 子系统连接检查

  检查子系统是否正在运行并能够响应请求。

- 子系统克隆连接和数据检查

  检查在给定 CS 子系统中配置的一组克隆的简单连接和数据健全性。给定的 CA 子系统的安全域被参考，以标识已设置的克隆。然后，检查会进入每个克隆，并在适用的情况下验证数据健全性。

22.2.1.2. PKI Healthcheck 配置

PKI Healthcheck 工具配置存储在 /etc/pki/healthcheck.conf 中。它类似如下：

```
[global]
    plugin_timeout=300
    cert_expiration_days=30

    # Dogtag specific section
    [dogtag]
    instance_name=pki-tomcat
```

22.2.1.3. 运行 PKI Healthcheck

- 要执行健康检查，请运行 pki-healthcheck 命令。
- 您还可以执行特定的检查。例如：

```
# pki-healthcheck --source pki.server.healthcheck.meta.csconfig --check
DogtagCertsConfigCheck
```

有关可能选项的更多信息，请参阅 man page: man pki-healthcheck。

22.2.1.4. 健康检查输出格式

健康检查会生成以下输出，您可以使用 --output-type 设置：

- 默认情况下，JSON 格式的机器可读输出(json)。
- 或者，人类可读的输出(人类可读)。

您可以使用 --output-file 选项指定替代文件目的地。

22.2.1.5. 健康检查结果

这个报告包含描述正在运行的内容和状态的消息。每个 Healthcheck 模块返回以下结果之一：

SUCCESS

按预期配置，检查已执行并发现没有问题

WARNING

不是错误，但值得注意正在进行或评估（例如，证书将很快过期）

ERROR

未按预期配置，但服务器可能仍然可以正常工作（例如，克隆冲突）

CRITICAL

未按预期配置，可能会有较大的影响（例如，服务没有启动，证书已过期等）。

如果状态不成功，则消息可能包含附加信息或重新命令，管理员可使用它们来更正问题（例如，文件具有错误的权限，预期 X 和获取 Y）。

部分 V. 参考

附录 A. 证书配置文件输入和输出参考

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 System 中的许多其他组件一样，配置文件输入和输出也作为 JAVA 插件实施，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引用。

- 第 A.1 节 “输入参考”
- 第 A.2 节 “输出参考”

A.1. 输入参考

输入会将某些字段放在与特定证书配置文件关联的注册页面中。为证书配置文件设置的输入用于动态使用适当的字段生成注册页面；这些输入字段收集配置文件以生成最终证书的必要信息。

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单的注册。它允许从下拉列表中选择请求格式，并提供输入字段来粘贴请求。

此输入会将以下字段置于注册表单中：

- 证书请求类型。此下拉菜单允许用户指定证书请求类型。选择是 PKCS the10 或 CRMF。PKCS11410 和 CRMF 支持通过加密消息语法(CMC)注册的证书管理消息。
- 证书请求.这是要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 证书请求输入用于使用 CMS (CMC)证书请求的证书消息注册，以请求表单提交。请求类型必须是 PKCS the10 或 CRMF，唯一的字段是要粘贴请求的 Certificate Request 文本区域。

例 A.2.

```
caCMCUserCert.cfg:input.i1.class_id=cmcCertReqInputImpl
```

A.1.3. 双密钥生成输入

双密钥生成输入是用于生成双密钥对的注册，因此签发两个证书，一个用于签名，另一个用于加密。

此输入会将以下字段置于注册表单中：

- 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.此字段在密钥生成请求中为密钥和证书设置密钥大小选择。

例 A.3.

```
caDualCert.cfg:input.i1.class_id=dualKeyGenInputImpl
```

A.1.4. 文件签发输入

File-Signing 输入设置为文件签名的字段，以显示该文件未被篡改。

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.此输入添加一个下拉菜单，以选择要在密钥生成请求中使用的密钥大小。
- URL Of 文件成为符号.这提供了要签名的文件的位置。

- 文本已签名.这提供了文件名。

例 A.4.

```
caAgentFileSigning.cfg:input.i2.class_id=fileSigningInputImpl
```

A.1.5. 镜像输入

Image input 设置字段来签署镜像文件。此输入创建的唯一字段是 镜像 URL，它提供了要签名的镜像的位置。

A.1.6. 密钥生成输入

Key Generation 输入用于生成单个密钥对的注册，其中通常基于用户的证书注册。

此输入会将以下字段置于注册表单中：

- 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.此输入添加一个下拉菜单，以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 输入用于注册硬件令牌的密钥，以便稍后用于基于证书的身份验证的代理。

此输入会将以下字段置于注册表单中：

- 令牌密钥 CUID.此字段为令牌设备提供 CUID （通常唯一的用户 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 输入用于注册硬件令牌用户的密钥，以便代理稍后使用令牌进行基于证书的身份验证。此输入会将以下字段置于注册表单中：

- 令牌密钥用户 UID.此字段提供令牌设备用户的 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

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。

这是需要与续订表单一起使用的唯一输入；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

主题 DN 输入允许用户输入特定的 DN 来设置为证书主题名称，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，主题名称输入用于注册。参数用于公式证书中的主题名称。此输入会将以下字段置于注册表单中：

- UID (LDAP 目录用户 ID)
- 电子邮件
- 通用名称 （用户的名称）
- 用户所属 组织单元（组织单元(ou)）
- 机构（机构名称 ）
- 国家 （用户所在的国家）

例 A.10.

```
caDualCert.cfg:input.i2.class_id=subjectNameInputImpl
```

A.1.12. 提交信息输入

提交信息输入收集证书请求者的信息，如名称、电子邮件和手机。

此输入会将以下字段置于注册表单中：

- 请求者名称

- *请求者电子邮件*
- *请求者电话*

例 A.11.

```
caAdminCert.cfg:input.i2.class_id=submitterInfoInputImpl
```

A.1.13. 通用输入

通用输入允许管理员指定与处理模式的扩展插件一起使用的任意数量的输入字段。例如，cc m 和 GUID 参数在模式主题替代名称扩展默认插件中使用：

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

Subject Alternative Name Extension Input 与 Subject Alternative Name Extension Default 插件一起使用。它允许管理员在 URI 中启用编号的参数，其模式为 req_san_pattern_ 在输入中，因此 SubjectAltNameExt 扩展。例如， URI 包括：

```
...&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=3
```

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

此输出显示以用户为print 格式显示证书。无法配置或更改此输出。它不以 pretty-print 格式显示证书以外的任何内容。

需要为任何自动注册指定此输出。用户使用自动注册方法成功验证后，证书会自动生成，并且此输出页面返回给用户。在代理批准的注册中，用户可以通过在终端实体页面中提供请求 ID 来获取证书（在签发后）。

例 A.14.

```
caAdminCert.cfg:output.o1.class_id=certOutputImpl
```

A.2.2. PKCS the7 输出

此输出返回 cert 和 certificate chain，格式为 PKCS the7。PKCS the7 格式是加密消息语法标准，用于签名。无法配置或更改此输出。

例 A.15.

```
caAgentFileSigning.cfg:output.o1.class_id=pkcs7OutputImpl
```

A.2.3. nsNSKeyOutput

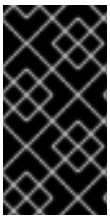
此类实施输出插件，用于返回令牌密钥的 DER 编码证书。

A.2.4. CMMF 输出

此输出返回证书管理消息格式(CMMF)中的证书。CMMF 管理 PKI 不同部分之间的通信，用于请求证书和请求证书撤销。

附录 B. 证书和 CRL 的默认值、约束和扩展

本附录解释了 X.509 v3 定义的标准证书扩展，以及 Netscape 定义的标准证书扩展，它们在完成 X.509 v3 之前发布的产品版本中使用的扩展。它为扩展提供特定类型证书的建议，包括 PKIX Part 1 建议。



重要

本附录是对使用或可在 Red Hat Certificate System 中配置的默认值、约束和 CRL 扩展的引用。有关证书和 CRL 扩展的完整参考和解释，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

默认用于定义证书的内容。本节列出并定义预定义的默认值。

B.1.1. 授权信息访问默认扩展

此默认附加授权信息访问扩展。此扩展指定应用程序如何验证证书如何访问信息，如在线验证服务和 CA 策略数据，关于已签发证书的 CA。此扩展不应用于直接指向 CA 维护的 CRL 位置；CRL 分发点扩展 第 B.1.7 节 “CRL 分发点扩展默认” 提供 CRL 位置的引用。

有关此扩展的一般信息，请参考 第 B.3.1 节 “authorityInfoAccess”。

此默认值可以定义以下限制：

- 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 “没有约束”。

此默认值最多可定义五个位置，每个位置的参数。参数在表中标有 n，以显示参数关联的位置。

表 B.1. 授权信息访问扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
Method_n	指定检索已发布扩展证书的 CA 的更多信息的访问方法。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) • 续订(2.16.840.1.113730.16.1)

参数	描述
LocationType_n	指定包含签发证书的 CA 的更多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	指定获取签发证书的 CA 的更多信息的地址或位置。 • 对于 directoryName，该值必须是 X.500 名称的字符串形式，类似于证书中的主题名称。例如： cn=SubCA, ou=Research Dept, o=Example company, c=US。 •

参数	描述
	对于 dNSName，该值必须是有效的完全限定域名。例如，testCA.example.com。 • 对于 EDIPartyName，该值必须是 IA5String。例如，公司示例。 • 对于 iPAddress，该值必须是有效的 IP 地址。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 对于 OID，该值必须是以点分开的数字组件表示法指定的唯一有效 OID。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，该值必须是非相对的通用资源标识符(URI)，遵循 URL 语法和编码规则。名称必须包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如：http://ocspResponder.example.com:8000。证书系统同时允许 IPv4 和 IPv6 IP 地址。
Enable_n	指定是否启用此位置。选择 true 来将其标记为 set；选择 false 来禁用它。

B.1.2. 授权密钥标识符扩展默认值

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。这个默认值没有参数。如果使用，则证书中包含此扩展以及公钥信息。

这个默认值采用以下约束：

- 无限制；请参阅 第 B.2.8 节 “没有约束”。

有关此扩展的一般信息，请参考 第 B.3.2 节 “authorityKeyIdentifier”。

B.1.3. 身份验证令牌主题名称默认值

此配置集默认根据身份验证令牌(AuthToken)对象中的属性值填充主题名称。

此默认插件可用于基于目录的身份验证管理器。基于目录的用户双用途证书注册证书配置文件有两个输入参数：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。

另外，基于目录的身份验证管理器对签发证书的主题名称进行公式。它使用来自 AuthToken 的用户 DN 值形成主题名称。

此默认负责从 AuthToken 读取主题名称，并将其放在证书请求中，以便最终证书包含主题名称。

此默认值可以定义以下限制：

- 无限制；请参阅 第 B.2.8 节 “没有约束”。

B.1.4. 基本限制扩展默认

此默认将 Basic Constraint 扩展附加到证书。扩展标识证书管理器是否为 CA。证书链验证过程中也会使用扩展来识别 CA 证书并应用证书链长度限制。

有关此扩展的一般信息，请参考 第 B.3.3 节 “basicConstraints”。

此默认值可以定义以下限制：

- 基本限制约束约束；请参阅 第 B.2.1 节 “基本约束约束”。
- 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 “没有约束”。

表 B.2. 基本限制扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
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，可按以下链的最大 CA 证书数（从属到）要发布的 CA 证书。路径长度会影响在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。 如果在最终用户证书中设置了扩展，则 maxPathLen 参数无效。 允许的值为 0 或 n。该值应小于 CA 签名证书的基本限制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；在路径中只能有一个最终用户证书。N 必须是大于零的整数。它指定在从属 CA 证书下允许的最大从属 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基本限制扩展中设置的值决定的值。如果签发者的路径长度没有限制，则从属 CA 证书中的路径长度也会不受限制。如果签发者的路径长度大于零的整数，则从属 CA 证书中的路径长度将设置为小于签发者路径长度的值；例如，如果签发者的路径长度为 4，则从属 CA 证书的路径长度将被设置为 3。

B.1.5. CA Validity 默认

此默认向 CA 证书注册或续订配置文件添加一个选项，以绕过 CA 的签名证书的过期约束。这意味着发布的 CA 证书可以有过期日期，其日期超过签发 CA 签名证书过期日期。

此默认值可以定义以下限制：

- 有效约束；请参阅 第 B.2.14 节 “有效约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

表 B.3. CA Validity 默认参数

参数	描述
bypassCAnotafterrange	设置请求 CA 是否可以请求其有效期超过发布 CA 有效周期的证书的默认值。
range	指定此证书的绝对有效期（以天数为单位）。
startTime	根据当前时间设置有效期何时开始。

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。此扩展定义了一个或多个策略，指示证书发布的策略以及可以使用证书的目的。此默认值定义最多五个策略，但可以是值。

有关此扩展的一般信息，请参考 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 来标记此扩展关键；选择 false 来标记非关键扩展名。
numCertPolicies	指定可以定义的策略数量。默认值为 5。
enable	选择 true 来启用策略；选择 false 来禁用该策略。
policyId	指定策略的 OID 标识符。
cpsURI.enable	扩展可以包含签发者证书实践声明的 URI。选择 true 来启用 URI；选择 false 来禁用 URI。

参数	描述
CPSURI.value	这个值是 CA 发布的认证实践声明(CPS)的指针。指针采用 URI 的形式。
usernotice.enable	扩展可以包含到签发者证书实践声明的 URI，也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	此可选用户通知参数是指向其他位置存储的消息的一系列数字。
usernotice.noticeReference.organization	此可选 user notice 参数指定公司的名称。
usernotice.explicitText.value	此可选 user notice 参数包含证书内的消息。

B.1.7. CRL 分发点扩展默认

此默认将 CRL 分发点扩展附加到证书。此扩展标识验证证书的应用程序可以从中获取 CRL 信息的位置，以验证证书的撤销状态。

有关此扩展的一般信息，请参考 第 B.3.5 节 “CRLDistributionPoints”。

此默认值可以定义以下限制：

- 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 “没有约束”。

此默认值定义最多五个位置，以及每个位置的参数。参数在表中标有 n，以显示参数关联的位置。

表 B.5. CRL 分发点扩展配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。

参数	描述
Type_n	指定 CRL 分发点的类型。可能的值有 DirectoryName、URIName 或 RelativeToIssuer。这个类型必须与 Name 字段中的值对应。
Name_n	指定 CRL 发布点的名称，名称可以是以下格式： • RFC 2253 语法中的 X.500 目录名称。名称类似于证书中的主题名称，如 cn=CA Central, ou=Research Dept, o=Example company, c=US。 • URIName，如 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 CRL 在分发点维护的吊销原因。提供以下常量的逗号分隔列表： • 未使用 • keyCompromise • cACompromise • affiliationChanged • 被取代 • cessationOfOperation • certificateHold

参数	描述
IssuerType_n	指定在发布点上为 CRL 维护签名的签发者的命名类型。签发者名称可以采用以下格式： • *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。可能的值如下：

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 名称的字符串形式，类似于证书中的主题名称。例如： cn=SubCA, ou=Research Dept, o=Example company, 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 EDIPartyName，该值必须是 IA5String。例如，公司示例。 • 对于 URIName，该值必须是 URL 语法和编码规则后的非相对 URI。名称必须包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如：http://testCA.example.com。证书系统支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，该值必须是有效的 IP 地址。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如， 0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 对于 OIDName，该值必须是以点分隔的数字组件表示法指定的唯一有效 OID。例如， 1.2 .3.4.55.6.5.99。 •

参数	描述
	otherName 用于任何其它格式的名称；这支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 格式为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。 otherName 必须具有 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. 扩展密钥用法扩展默认值

默认情况下，将扩展的密钥用法扩展附加到证书。

有关此扩展的一般信息，请参考 第 B.3.6 节 “extKeyUsage”。

除了 Key Usage 扩展中指示的基本用途外，扩展还标识了使用认证的公钥。例如，如果密钥使用扩展标识了签名密钥，则扩展的密钥使用扩展可缩减密钥的使用，以仅签署 OCSP 响应或只有 Java™ 小程序。

表 B.6. 用于扩展密钥用法扩展的 PKIX 使用定义

使用方法	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

使用方法	OID
IPsec 端到端系统	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 可以使用包含以下两个 OID 的扩展密钥用法扩展的证书加密硬盘上的文件，称为加密的文件系统(EFS)：

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)

当用户丢失私钥且需要使用该密钥加密的数据时，恢复代理会使用 EFS 恢复证书。证书系统支持这两个 OID，并允许签发包含这些 OID 的扩展密钥用法扩展的证书。

普通用户证书应该只使用 EFS OID 创建，而不是恢复 OID。

此默认值可以定义以下限制：

- 扩展键使用约束；请参阅 第 B.2.3 节 “扩展键使用约束”。
- 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 “没有约束”。

表 B.7. 扩展密钥用法扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
OID	指定标识键使用目的的 OID。允许的数值是唯一的、由点分开的数字组件标记中指定的有效 OID。例如： 2.16.840.1.113730.1.99。根据键使用目的，OID 可以被 PKIX （在 表 B.6 “用于扩展密钥用法扩展的 PKIX 使用定义”中列出）或自定义 OID 指定。自定义 OID 必须位于为公司使用保留的 ID 的注册子树中。虽然可以使用自定义 OID 来评估和测试证书系统，但在生产环境中，遵守 ISO 规则来定义 OID 和 ID 的注册子树。

B.1.9. Newest CRL 扩展默认值

此默认将 Freshest CRL 扩展附加到证书。

此默认值可以定义以下限制：

- 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 “没有约束”。

此默认定义五个位置，以及每个位置的参数。参数在表中标有 n，以显示参数关联的位置。

表 B.8. 最旧的 CRL 扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
PointEnable_n	选择 true 来启用此点；选择 false 来禁用此点。
PointType_n	指定发布点的类型，可以是 DirectoryName 或 URIName。

参数	描述
PointName_n	• 如果将 pointType 设为 directoryName，则该值必须是 X.500 名称，类似于证书中的主题名称。例如： cn=CACentral,ou=Research Dept,o=Example company,c=US。 • 如果 pointType 设为 URIName，则名称必须是 URI，则指定主机的绝对路径。例如： http://testCA.example.com/get/crls/here/。
PointIssuerName_n	指定签名 CRL 的签发者名称。名称可以是以下格式： • 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 名称的字符串形式，类似于证书中的主题名称。例如： cn=SubCA, ou=Research Dept, o=Example company, 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 EDIPartyName，该值必须是 IA5String。例如，公司示例。

参数	描述
	• 对于 URIName，该值必须是 URL 语法和编码规则后的非相对 URI。名称必须包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如：http://testCA.example.com。证书系统支持 IPv4 和 IPv6 地址。 • 对于 IPAddress，该值必须是有效的 IP 地址。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 对于 OIDName，该值必须是以点分隔的数字组件表示法指定的唯一有效 OID。例如，1.2 .3.4.55.6.5.99。 • otherName 用于任何其它格式的名称；这支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 格式为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。 otherName 必须具有 格式 （类型）oid，字符串。例如，(IA5String) 1.2.3.4,MyExample。 name 值必须符合 PointType_ 中指定的格式。

参数	描述
PointType_n	指定为 CRL 签名的 CRL 签发者的一般名称类型。可能的值如下： • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • ipaddress • OIDName • OtherName 此参数的值必须与 PointIssuerName 字段中的值对应。

B.1.10. 通用扩展默认值

此扩展允许使用用户确定的数据创建通用扩展。默认可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 来标记此扩展关键；选择 false 来标记非关键扩展名。
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签发到 CA 的证书。禁止任何策略表示特殊 anyPolicy OID，值为 { 2 5 29 32 0 }，不被视为其他证书策略的显式匹配。

表 B.10. 禁止任何策略扩展默认配置参数

参数	描述
Critical	此策略必须标记为关键策略。选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
SkipCerts	此参数指定在不再允许任何策略前可能出现在路径中的额外证书数量。值 1 表示可在该证书主题发布的证书中处理任何策略，但不能在路径中的其他证书中处理。

B.1.12. 签发者备用名称扩展默认

此默认将 Issuer Alternative Name 扩展附加到证书。Issuer Alternative Name 扩展用于将互联网风格的身份与证书签发者关联。

此默认值可以定义以下限制：

- 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 “没有约束”。

此默认定义五个位置，以及每个位置的参数。参数在表中标有 n，以显示参数关联的位置。

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** 来标记此扩展关键；选择 **false** 来标记非关键扩展名。
issuerAltExtType	这会设置要使用的名称扩展类型，可以是以下之一： • **RFC822Name** • **DirectoryName** • **DNSName** • **EDIPartyName** • **URIName** • **ipaddress** • **OIDName**

参数	描述
issuerAltExtPattern	指定要包含在扩展中的 request 属性值。属性值必须符合任何受支持的通用名称类型。permissable 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，它会在扩展中设置属性值，并将扩展添加到证书。如果指定了多个属性，且请求中不存在任何属性，则服务器不会向证书添加 Issuer Alternative Name 扩展。如果没有从请求中使用合适的属性来组成 issuerAlternativeName，那么可在没有任何令牌表达式的情况下使用字面字符串。例如，证书颁发机构。

B.1.13. 密钥使用扩展默认值

此默认将 Key Usage 扩展附加到证书。扩展指定应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，这限制了密钥对的使用为预先确定的目的。

有关此扩展的一般信息，请参考 第 B.3.8 节 “keyUsage”。

此默认值可以定义以下限制：

- 关键使用约束；请参阅 第 B.2.6 节 “关键使用扩展约束”。
- 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 “没有约束”。

表 B.12. 密钥使用扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
digitalSignature	指定是否允许签名 SSL 客户端证书和 S/MIME 签名证书。选择要设置的 true。
nonRepudiation	指定是否用于 S/MIME 签名证书。选择要设置的 true。  **WARNING** 使用这个位是相似的。在为任何证书设置前，请仔细考虑其使用的法律后果。
keyEncipherment	指定主题中的公钥是否用于增强私钥或 secret 密钥。这是为 SSL 服务器证书和 S/MIME 加密密钥设置的。选择要设置的 true。
dataEncipherment	指定在使用主题的公钥而不是密钥材料时，是否设置扩展。选择要设置的 true。
keyAgreement	指定是否在主题的公钥用于密钥协议时设置扩展。选择要设置的 true。
keyCertsign	指定是否使用公钥来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 CRL 的 CA 签名证书设置扩展。选择要设置的 true。
encipherOnly	指定如果公钥仅在执行密钥协议时加密数据时，是否设置扩展。如果设置了此位，则还应设置 keyAgreement。选择要设置的 true。

参数	描述
decipherOnly	指定在执行密钥协议时，如果公钥仅用于解密数据，则指定是否设置扩展。如果设置了此位，则还应设置 keyAgreement。选择要设置的 true。

B.1.14. 名称限制扩展默认

此默认为证书附加一个 Name Constraints 扩展。CA 证书中使用扩展来指示证书链中后续证书中的主题名称或主题备用名称应位于的名称。

有关此扩展的一般信息，请参考 第 B.3.9 节 “nameConstraints”。

此默认值可以定义以下限制：

- 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 “没有约束”。

此默认值为允许子树和排除的子树定义最多五个位置，并为每个位置设置参数。参数在表中标有 n，以显示参数关联的位置。

表 B.13. 名称限制扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。

参数	描述
PermittedSubtreesn.min	指定允许的最小子树数量。 • -1 指定不应在扩展中设置该字段。 • 0 指定最小子树数为零。 • N 必须是大于零的整数。它设置所需的子树数量。
PermittedSubtreesmax_n	指定允许子树的最大数量。 • -1 指定不应在扩展中设置该字段。 • 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	指定要包含在扩展名中的允许子树的通用名称类型。可能的值如下： • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • ipaddress • OIDName • OtherName
PermittedSubtreeNameValue_n	指定要包含在扩展名中的允许子树的常规名称值。 • 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 名称的字符串形式，类似于证书中的主题名称。例如： cn=SubCA, ou=Research Dept, o=Example company, 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 EDIPartyName，该值必须是 IA5String。例如，公司示例。
	• 对于 URIName，该值必须是 URL 语法和编码规则后的非相对 URI。名称必须包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如： http://testCA.example.com。证书系统支持 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.n 格式，或使用子网掩码的 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:fe56:4495/64 或 2001:db8::/64。
	• 对于 OIDName，该值必须是以点分隔的数字组件表示法指定的唯一有效 OID。例如， 1.2 .3.4.55.6.5.99。
	• otherName 用于任何其它格式的名称；这支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 格式

参数	描述
	为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。 otherName 必须具有 格式 （类型）oid，字符串。例如，(IA5String) 1.2.3.4,MyExample。
PermittedSubtreeEnable_n	选择 true 来启用此允许的子树条目。
ExcludedSubtreesn.min	指定排除的子树的最小数量。 • -1 指定不应在扩展中设置该字段。 • 0 指定最小子树数为零。 • N 必须是大于零的整数。这会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 指定不应在扩展中设置该字段。 • 0 指定子树的最大数量为零。 • N 必须是大于零的整数。这会设置允许的最大子树数。

参数	描述
ExcludedSubtreeNameChoice_n	指定要包含在扩展中的未排除子树的一般名称类型。可能的值如下： • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • ipaddress • OIDName • OtherName
ExcludedSubtreeNameValue_n	指定要包含在扩展名中的允许子树的常规名称值。 • 对于 RFC822Name，该值必须是有效的互联网电子邮件地址。例如：

参数	描述
	testCA@example.com。 • 对于 DirectoryName，该值必须是 X.500 名称，类似于证书中的主题名称。例如： cn=SubCA, ou=Research Dept, o=Example company, 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 EDIPartyName，该值必须是 IA5String。例如，公司示例。 • 对于 URIName，该值必须是 URL 语法和编码规则后的非相对 URI。名称必须包含方案，如 http，以及主机的完全限定域名或 IP 地址。例如： http://testCA.example.com。证书系统支持 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.n 格式，或使用子网掩码的 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:fe56:4495/64 或 2001:db8::/64。 • 对于 OIDName，该值必须是以点分隔的数字组件表示法指定的唯一有效 OID。例如，1.2 .3.4.55.6.5.99。 • 对于 OtherName，值是带有任何其他格式的名称。这支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 格式为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。

参数	描述
	otherName 必须具有 格式 （类型）oid，字符串。例如，(IA5String) 1.2.3.4,MyExample。
ExcludedSubtreeEnable_n	选择 true 来启用此排除的子树条目。

B.1.15. Netscape 证书类型扩展默认

WARNING

这个扩展已过时。改为使用 Key Usage 或 Extended Key Usage 证书扩展。

此默认将 Netscape 证书类型扩展附加到证书。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。这限制了将证书的使用为预先确定的目的。

B.1.16. Netscape comment Extension Default

WARNING

这个扩展已过时。

此默认为证书附加 Netscape 注释扩展。扩展可用于在证书中包含文本注释。在使用或查看证书时，可以解释注释的应用程序会显示它。

有关此扩展的常规信息，请参考 第 B.4.3.2 节 “Netscape-comment”。

此默认值可以定义以下限制：

- 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 “没有约束”。

表 B.14. Netscape 注释扩展配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
CommentContent	指定要出现在证书中的注释内容。

B.1.17. 没有默认扩展

在不使用默认值时，可以使用此默认值来设置限制。这个默认值没有设置，且没有设置默认值，但允许设置所有约束。

B.1.18. OCSP 没有检查默认扩展

这个默认为证书附加一个 OCSP No Check 扩展。这个扩展只在 OCSP 响应器证书中使用，指示与 OCSP 兼容的应用程序如何验证授权 OCSP 响应的证书的撤销状态。

有关此扩展的一般信息，请参考 第 B.3.10 节 “OCSPNocheck”。

此默认值可以定义以下限制：

- 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 “没有约束”。

表 B.15. OCSP 没有检查扩展的默认配置参数

参数	描述

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。

B.1.19. 策略限制默认

此默认为证书附加策略约束扩展。扩展，它只可用于 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。默认值可同时指定 ReqExplicitPolicy 和 InhibitPolicyMapping。PKIX 标准要求证书中存在时，扩展不得由空序列组成。必须至少有两个指定字段之一。

有关此扩展的一般信息，请参考 第 B.3.11 节 “policyConstraints”。

此默认值可以定义以下限制：

- 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 “没有约束”。

表 B.16. 策略限制扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。

参数	描述
reqExplicitPolicy	指定在需要显式策略前在路径中允许的证书总数。这是在需要可接受的策略前，可以在从属 CA 证书下串联的 CA 证书数量。 • -1 指定不应在扩展中设置该字段。 • 0 指定在需要显式策略前，路径中不允许使用从属 CA 证书。 • N 必须是大于零的整数。它指定在需要显式策略前路径中允许的最大从属 CA 证书数。 这个数字会影响在证书验证过程中使用的 CA 证书数量。链以验证的最终证书开始，并移动链。如果在最终用户证书中设置扩展，该参数无效。

参数	描述
inhibitPolicyMapping	指定不再允许策略映射前允许的证书总数。 • -1 指定不应在扩展中设置该字段。 • 0 指定在不再允许策略映射前的路径中不允许从属 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的路径中允许的最大从属 CA 证书数。例如，1 表示策略映射可以在该证书主题发布的证书中处理，但不能在路径中的其他证书中处理。

B.1.20. 策略映射器默认扩展

此默认将策略映射扩展附加到证书。扩展列表 OID 对，每个对标识两个 CA 的两个策略语句。对表示一个 CA 的对应策略等同于另一个 CA 的策略。扩展在跨交互的上下文中可能很有用。如果支持，则扩展仅包含在 CA 证书中。一个 CA 的默认策略语句通过对分配给其策略语句的 OID 相互映射

每个对由两个参数定义，即 issuerDomainPolicy 和 subjectDomainPolicy。pairing 表示发布 CA 认为与主题 CA 的 subjectDomainPolicy 等效的 issuerDomainPolicy。发布 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 关联的策略等同于它们接受的策略。

有关此扩展的一般信息，请参考 第 B.3.12 节 “policyMappings”。

此默认值可以定义以下限制：

- 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 “没有约束”。

表 B.17. 策略映射扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。
IssuerDomainPolicy_n	指定分配给发布 CA 的 policy 语句的 OID，以使用另一个 CA 的 policy 语句进行映射。例如，1.2 .3.4.5。
SubjectDomainPolicy_n	指定分配给主题 CA 的策略声明的 OID，对应于发布 CA 的策略声明。例如：6.7.8.9.10。

B.1.21. 私钥使用周期默认扩展

Private Key Usage Period 扩展允许证书签发者为私钥指定与证书本身不同的有效期周期。此扩展用于数字签名密钥。

表 B.18. 私钥使用间隔配置参数

参数	描述
Critical	此扩展应该始终不是关键的。
puStartTime	此参数设置开始时间。默认值为 0，从激活扩展时开始有效期。
puDurationDays	此参数设置使用周期的持续时间。默认值为 365，它会从激活扩展的时间将有效期设置为 365 天。

B.1.22. 签名算法默认

此默认在证书请求中附加了一个签名算法。此默认提供了一个代理，它带有可能算法，可用于签署证

书。

此默认值可以定义以下限制：

- 签名算法约束；请参阅 第 B.2.10 节 “签名算法约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

表 B.19. 签名算法默认配置参数

参数	描述
signingAlg	指定用于创建此证书的默认签名算法。代理可以通过指定 signingAlgsAllowed 参数中包含的一个值来覆盖这个值。
signingAlgsAllowed	指定可用于签署此证书的签名算法。该算法可以是以下任何或全部： • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA

B.1.23. 主题备用名称扩展默认值

此默认为证书附加一个 Subject Alternative Name 扩展。扩展会将其他身份（如电子邮件地址、DNS 名称、IP 地址(IPv4 和 IPv6)或 URI 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，则 Subject Alternative 名称扩展必须包含主题的替代名称，且扩展被标记为 critical。

对于任何基于目录的身份验证方法，证书系统可以检索任意字符串和字节属性的值，并在证书请求中设置它们。这些属性可以通过在 IdapStringAttributes 和 IdapByteAttributes 字段中输入自动注册模块中定义的 IdapByteAttributes 字段来设置它们。

如果经过身份验证的用户 - 表示存储在 LDAP 数据库中的属性 - 需要是此扩展的一部分，请使用 $request 中的值。X$ 令牌。

有一个额外的属性，可将通用唯一标识符(UUID)插入到主题 alt 名称中。这个选项为版本 4 UUID 生成随机数字；模式是通过引用服务器来定义的，该服务器将在额外的 subjAltExtSource 参数中生成数字。

在示例中配置了基本 Subject Alternative Name 扩展默认。

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.c
om
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=5
```

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，配置集会读取其值并在扩展中设置它。如果配置了基于 LDAP 的身份验证，则 Subject Alternative Name 扩展默认可以从 LDAP 目录中插入属性值。添加到证书的扩展包含所有配置的属性。

表 B.20 “主题备用名称中的值的变量” 中列出了可与 Subject Alternative Name 扩展默认一起使用的

变量。

表 B.20. 主题备用名称中的值的变量

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(mail)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。这可以是 IPv4 或 IPv6 地址，具体取决于客户端。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。
$request.profileRemoteHost$	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 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。

策略设置令牌	描述
$request.requestor_email$	提交请求的人员的电子邮件地址。
$request.requestowner$	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令牌的唯一 ID (CUID)。
$request.upn$	Microsoft UPN。它具有格式 (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 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 参数控制需要将列出的属性添加到证书中。这个参数必须添加到自定义配置集中，可能需要修改默认配置集，以便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 5 以插入 RFC822Name、DNSName、URIName、OtherName, 和 RFC822Name 名称（通用名称 _0、_1、_2、_3、_4）。

此默认值可以定义以下限制：

- 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 “没有约束”*。

表 B.21. 主题备用名称扩展默认配置参数

参数	描述
Critical	选择 true 来标记此扩展关键；选择 false 来标记非关键扩展名。

参数	描述
pattern	指定要包含在扩展中的 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，它会在扩展中设置属性值，并将扩展添加到证书。如果指定了多个属性，且请求中不存在任何属性，则服务器不会向证书添加 Subject Alternative Name 扩展。permissable 值是证书请求中包含的 request 属性。例如，$request.requestor_email$。
类型	指定 request 属性的常规名称类型。 • 如果 request-attribute 值是 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 company, 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 EDIparty 名称，请选择 EDIPartyName。例如，公司示例。 • 如果 request-attribute 值是一个包含方案的非相对 URI，如 http，以及主机的完全限定域名或 IP 地址，请选择 URIName。例如：http://hr.example.com。证书系统支持 IPv4 和 IPv6 地址。

参数	描述
	• 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 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 如果 request-attribute 值是唯一的、由点分隔的数字组件标记中指定的有效 OID，请选择 OIDName。例如，1.2 .3.4.55.6.5.99。 • 为名称选择 OtherName，使用任何其他格式。这支持 PrintableString、IA5String、UTF8String、BMPString、任何、和 KerberosName。KerberosName 格式为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。 otherName 必须具有 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。唯一支持的源是 UUID4，它会生成一个随机数字来创建 UUID。
组件数(NumGNs)	指定必须包含在主题备用名称中的名称组件数量。

B.1.24. 主题目录属性扩展默认值

此默认将 Subject Directory 属性扩展附加到证书。Subject Directory Attributes 扩展处理证书的主题所需的目录属性值。

此默认值可以定义以下限制：

- 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 “没有约束”。

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 来标记此扩展关键；选择 false 来标记非关键扩展名。
Name	属性名称；可以是任何 LDAP 目录属性，如 cn 或 mail。
pattern	指定要包含在扩展中的 request 属性值。属性值必须符合属性允许的值。如果服务器找到属性，它会在扩展中设置属性值，并将扩展添加到证书。如果指定了多个属性，且请求中不存在任何属性，则服务器不会向证书添加 Subject Directory Attributes 扩展。例如，$request.requestor_email$。
启用	设置该属性是否可以添加到证书中。选择 true 来启用属性。

B.1.25. 主题信息访问默认扩展

实施注册默认策略，在证书模板中填充 Subject Information Access 扩展。此扩展表示如何访问显示扩展的证书主题的信息和服务。

参数	描述
Critical	此扩展应该是非关键的。
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	访问方法的类型。 • URIName • 目录名称 • DNS 名称 • EID Party 名称 • IP 地址 • OID 名称 • RFC822Name
subjInfoAccessADLocation_n	基于类型 subjInfoAccessADMethod_n 的位置 例如，URI Name 的 URL。

参数	描述
subjInfoAccessADEnable_n	选择 true 来启用此扩展；选择 false 来禁用此扩展。

B.1.26. 主题密钥标识符默认

默认情况下，为证书附加一个 Subject Key Identifier 扩展。扩展标识包含特定公钥的证书，它标识具有相同主题名称的多个证书。

有关此扩展的一般信息，请参考 第 B.3.16 节 “subjectKeyIdentifier”。

如果启用，如果扩展尚不存在，配置集会将 Subject Key Identifier Extension 添加到注册请求中。如果请求中存在扩展，如 CRMF 请求，则默认会替换扩展。代理批准手动注册请求后，配置集接受尚未有的任何 Subject Key Identifier Extension。

这个默认值没有参数。如果使用，则证书中包含此扩展以及公钥信息。

此默认值可以定义以下限制：

- 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 “没有约束”。

B.1.27. 主题名称默认

默认情况下，将服务器端可配置的主题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。

此默认值可以定义以下限制：

- 主题名称约束；请参阅 第 B.2.11 节 “主题名称约束”。

- 唯一主题名称约束；请参阅 第 B.2.13 节 “唯一主题名称约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

表 B.23. 主题名称默认配置参数

参数	描述
Name	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，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

默认情况下，将用户提供的密钥附加到证书请求中。这是必需的默认值。密钥是注册请求的一部分。

此默认值可以定义以下限制：

- 密钥约束；请参阅 第 B.2.5 节 “键约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

B.1.29. 用户签名算法默认

此默认实施一个注册默认配置文件，它会在证书请求中填充用户提供的签名算法。如果证书配置文件中包含，这允许用户为证书选择签名算法，受约束集约束。

不提供输入来将签名算法选择添加到注册表单中，但可以提交包含此信息的请求。

此默认值可以定义以下限制：

- 签名算法约束；请参阅 第 B.2.10 节 “签名算法约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

B.1.30. 用户主题名称默认

默认情况下，将用户提供的主题名称附加到证书请求。如果证书配置文件中包含，它将允许用户为证书提供主题名称，受限制集的限制。此扩展会保留签发证书时在原始证书请求中指定的主题名称。

此默认值可以定义以下限制：

- 主题名称约束；请参阅 第 B.2.11 节 “主题名称约束”。
- 唯一主题名称约束；请参阅 第 B.2.13 节 “唯一主题名称约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

B.1.31. 用户验证器默认值

默认情况下，将用户提供的有效期附加到证书请求。如果证书配置文件中包含，它将允许用户提供有效期，受约束集约束。在签发证书时，此默认配置集会在原始证书请求中保留用户定义的有效期周期。

不提供输入来向注册表单添加用户提供的有效期日期，但可以提交包含此信息的请求。

此默认值可以定义以下限制：

- 有效约束；请参阅 第 B.2.14 节 “有效约束”。

- 无限制；请参阅 第 B.2.8 节 “没有约束”。

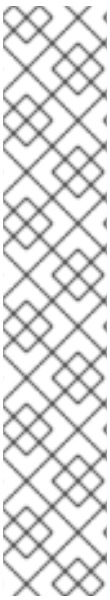
B.1.32. 用户 Supplied 扩展默认

User Supplied Extension Default 类使用证书请求中定义的任何证书扩展填充证书。这要求用户提交符合某些标准或授予某些信息的证书请求，因为配置集在注册证书前可能需要特定的扩展。

WARNING

值得注意的是设置此扩展默认，因为它允许用户在证书请求中指定扩展。如果使用此默认值，红帽强烈建议您使用与扩展对应的约束，以最小化用户 Supplied Extension Default 可能的任何可能。

用户定义的扩展会根据设定的任何约束进行验证，因此可以限制扩展类型（通过扩展约束），或者为密钥和其他基本限制设置规则，如这是 CA 证书。



注意

如果在具有对应 OID （扩展约束）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 Supplie d Extension Default 启用证书配置文件，则必须编辑此配置集来支持用户提供证书请求的扩展。应用 userExtensionDefaultImpl 默认，如示例所示。给定的 OID 用于基本约束约束。

```
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 通过三种方法之一通过用户 Supplied Extension 默认处理注册：

- 如果在证书请求和默认值中指定扩展的 OID，则扩展将由约束验证，并应用到证书。
-

如果请求中给出了扩展的 OID，但没有在配置集中的 User Supplied Extension Default 中指定，则用户指定的扩展将被忽略，并且证书成功注册没有该扩展。

- 如果在具有对应 OID （扩展约束）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，证书注册表单没有任何输入字段供用户添加用户提供的扩展。提交证书请求而不提供扩展失败。

例 B.2 “用于扩展密钥用法扩展的用户 Supplied Extension 默认” 使用扩展的密钥用法约束将用户扩展默认添加到配置集中。userExtOID 参数中指定的 OID 用于扩展密钥使用情况。

例 B.2. 用于扩展密钥用法扩展的用户 Supplied Extension 默认

```
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 “用于扩展密钥用法扩展的用户 Supplied Extension 默认” 中，虽然用户 Supplied Extension Default 允许用户指定 Extended Key Usage Extension (2.5.29.37)，但约束将用户请求限制为仅 SSL 客户端身份验证(1.3.6.1.5.5.7.3.2)和电子邮件保护(1.3.6.1.5.5.7.3.4)。

第 3.2 节 “设置证书配置文件” 中描述了编辑配置集。

例 B.3. CSR 中的多个用户 Supplied 扩展

RHCS 注册配置集框架允许在同一配置文件中定义多个用户 Supplied 扩展。例如，可以指定以下内容的组合：

- 对于扩展的密钥用法扩展：

```
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
```

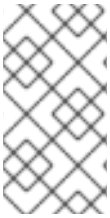
- *对于密钥用法扩展：*

  通过使用以下格式，您可以应用扩展参数的策略：

  - *必须存在于 CSR: value = "true"中*
  - *不得 存在于 CSR: value = "false"中*
  - *可选: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 创建带有用户定义的扩展的 CSR”。

B.1.33. 有效期默认值

默认情况下，将服务器端可配置的有效期附加到证书请求中。

此默认值可以定义以下限制：

- 有效约束；请参阅 第 B.2.14 节 “有效约束”。
- 无限制；请参阅 第 B.2.8 节 “没有约束”。

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何时开始。

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。本节列出了带有每个完整定义的预定义约束。

B.2.1. 基本约束约束

Basic Constraints 扩展约束检查证书请求中的基本约束是否满足此约束中设置的条件。

表 B.25. 基本约束约束配置参数

参数	描述
basicConstraintsCritical	指定扩展是否可以标记为 critical 还是非关键。选择 true 来标记这个扩展关键；选择 false 以防止此扩展标记为关键。选择一个连字符 -，代表没有严重首选项。
basicConstraintsIsCA	指定证书主题是否为 CA。选择 true 来为此参数要求值为 true （是 CA；为此参数选择 false 以禁止为 true 值；选择连字符 - 来表示没有为此参数放置任何限制。

参数	描述
basicConstraintsMinPathLen	指定允许的最少路径长度，其最大 CA 证书可以串联到以下的 CA 证书数（从属到）被签发的 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。 如果在最终用户证书中设置扩展，则此参数无效。 允许的值为 0 或 n。该值必须小于 CA 签名证书的基本限制扩展中指定的路径长度。 0 指定在发布的子 CA 证书下不允许从属 CA 证书；在路径中只有最终用户证书可能会遵循。 N 必须是大于零的整数。这是使用下级 CA 证书下允许的下级 CA 证书的最小数量。

参数	描述
basicConstraintsMaxPathLen	指定最大允许的路径长度、可发布的 CA 证书的最大数量（从属到）被签发的 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。 如果在最终用户证书中设置扩展，则此参数无效。 允许的值为 0 或 n。该值必须大于 CA 签名证书的基本限制扩展中指定的路径长度。 0 指定在发布的子 CA 证书下不允许从属 CA 证书；在路径中只有最终用户证书可能会遵循。 N 必须是大于零的整数。这是使用下级 CA 证书下允许的最大从属 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基本限制扩展设置的路径长度决定的值。如果签发者的路径长度没有限制，则从属 CA 证书中的路径长度也没有限制。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小于签发者的路径长度的值；例如，如果签发者的路径长度为 4，则从属 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 Validity 约束检查证书模板中的有效性周期是否在 CA 的有效周期内。如果证书的有效期不在 CA 证书的有效期外，则约束将被拒绝。

B.2.3. 扩展键使用约束

扩展的密钥用法扩展约束检查证书上的扩展密钥用法扩展是否满足此约束中设置的条件。

表 B.26. 扩展键用法扩展约束配置参数

参数	描述
exKeyUsageCritical	当设置为 true 时，扩展可以标记为关键。当设置为 false 时，扩展可以标记为非关键。
exKeyUsageOIDs	指定用于标识键使用目的的允许的 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

此约束实现了常规扩展约束。它检查是否存在扩展。

表 B.27. 扩展约束

参数	描述
extCritical	指定扩展是否可以标记为 critical 还是非关键。选择 true 来标记扩展关键；选择 false 来将其标记为非关键。select - 不强制首选项。
extOID	必须存在于证书中的扩展 OID 才能传递约束。

B.2.5. 键约束

此约束检查 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，KeyParameters 参数包含以逗号分隔的法律密钥大小列表，而 EC Keys the KeyParameters 参数包含以逗号分隔的可用 ECC curves 列表。

表 B.28. 关键约束配置参数

参数	描述
keyType	提供一个密钥类型；默认情况下，它被设置为 -，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被系统标识，则约束将被拒绝。
KeyParameters	定义特定的密钥参数。为键设置的参数不同，具体取决于 keyType 参数的值（根据键类型而定）。 • 使用 RSA 密钥时，KeyParameters 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. 关键使用扩展约束

Key Usage 扩展约束检查证书请求中的密钥使用量约束是否满足此约束中设置的条件。

表 B.29. 关键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 来将其标记为非关键。select - 代表无首选。
keyUsageDigitalSignature	指定是否签署 SSL 客户端证书和 S/MIME 签名证书。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。

参数	描述
kleyUsageNonRepudiation	指定是否设置 S/MIME 签名证书。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。  WARNING 使用这个位是相似的。在为任何证书设置前，请仔细考虑其使用的法律后果。
keyEncipherment	指定是否为 SSL 服务器证书和 S/MIME 加密密钥设置扩展。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。
keyUsageDataEncipherment	指定在使用主题的公钥加密用户数据而不是密钥材料时，是否设置扩展。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。
keyUsageKeyAgreement	指定是否在主题的公钥用于密钥协议时设置扩展。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。
keyUsageCertsign	指定扩展是否适用于所有 CA 签名证书。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。
keyUsageCRLSign	指定是否为用于签署 CRL 的 CA 签名证书设置扩展。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。

参数	描述
keyUsageEncipherOnly	指定是否在将公钥用于加密数据时设置扩展。如果设置了此位，则还应设置 keyUsageKeyAgreement。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。
keyUsageDecipherOnly	指定是否在将公钥用于解码数据时设置扩展。如果设置了此位，则还应设置 keyUsageKeyAgreement。选择 true 来将其标记为 set；选择 false 以保持此设置；选择连字符 - 来指示没有为此参数放置任何限制。

B.2.7. Netscape 证书类型扩展约束

WARNING

这个约束已过时。使用 Key Usage 扩展或扩展的密钥用法扩展，而不是使用 Netscape 证书类型扩展。

Netscape Certificate Type 扩展约束检查证书请求中的 Netscape Certificate Type 扩展是否满足此约束中设置的条件。

B.2.8. 没有约束

此约束没有实施约束。当与默认值一起选择时，在此默认值上没有限制。

B.2.9. 续订周期限制

当用户可以根据其过期日期续订证书时，Renewal Grace Period Constraint 会设置规则。例如，用户无法在证书过期前或其过期时间过期前更新证书。

使用此约束时要记住的一个事项是，此约束是在 原始注册配置集 上设置的，而不是续订配置集。续订宽限期的规则是原始证书的一部分，并针对后续续订执行并应用。

此约束仅适用于 No Default 扩展。

表 B.30. 续订周期约束配置参数

参数	描述
renewal.graceAfter	在证书过期后设置可以提交续订的周期（以天为单位）。如果证书已过期了这一时间，则续订请求将被拒绝。如果没有给出值，则没有限制。
renewal.graceBefore	设置在可以提交以进行续订的证书过期前的天数（以天为单位）。如果证书不是接近其过期日期，则拒绝续订请求。如果没有给出值，则没有限制。

B.2.10. 签名算法约束

Signing Algorithm 约束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. 签名算法约束配置参数

参数	描述

参数	描述
signingAlgsAllowed	设置可以指定签署证书的签名算法。该算法可以是以下任何或全部： • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA • SHA256withEC • SHA384withEC • SHA512withEC

B.2.11. 主题名称约束

Subject Name 约束检查证书请求中的主题名称是否满足条件。

表 B.32. 主题名称约束配置参数

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 Java 工具匹配，以匹配正则表达式。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）搜索任意数量的字符和句点(.)来搜索任何类型的字符。

例如，如果主题名称约束的模式设置为 uid=rhcs，则证书配置文件框架将检查证书请求中的主题名称是否与模式匹配。如 uid=user, o=Example, c=US ）的主题名称可以满足模式 uid= the。主题名称 cn=user, o=example,c=US 不满足模式。uid= 114 表示主题名称必须以 uid 属性开头；句点分隔(IANA)通配符允许遵循 uid 的任何类型和数字。

可以要求内部模式，如 Evolution ou=Engineering，这需要在之前和之后带有任何类型的字符串的 ou=Engineering 属性。这与 cn=jdoe,ou=internal,ou=west coast,ou=engineering,o="Example Corp",st=NC 和 uid=bjensen,ou=engineering,dc=example,dc=com 匹配。

最后，也可以通过在选项之间设置管道符号(|)来允许是一个字符串或另一个字符串的请求。例如，要允许包含 ou=engineering ,ou=body 或 ou=engineering, o="Example Corp" 的主题名称，其模式为 114 ou=engineering,ou=body \| Evolutionou=engineering,o="Example Corp"。

注意

对于使用特殊字符（如句点）的模式，请使用 back 斜杠(\)转义字符。例如，若要搜索字符串 o="Example Inc."，请将模式设置为 o="Example Inc\."。

证书请求中的主题名称和 UID 或 CN

用于构建主题 DN 的模式也可以基于请求证书的人员的 CN 或 UID。Subject Name Constraint 设置 CN （或 UID）的 patter 来识别证书请求的 DN，然后该 CN 上的 Subject Name Default 构建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. 唯一键约束

此约束检查公钥是否是唯一的。

表 B.33. 唯一的键约束参数

参数	描述
allowSameKeyRenewal	请求被视为续订，如果此参数设为 true（如果公钥不唯一），则接受它，如果主题 DN 与现有证书匹配。但是，如果公钥是重复的，且与现有 Subject DN 不匹配，则请求将被拒绝。 当参数设为 false 时，重复的公钥请求将被拒绝。

B.2.13. 唯一主题名称约束

unique Subject Name 约束限制服务器使用相同的主题名称发布多个证书。提交证书请求时，服务器会自动针对其他发布的证书别名检查 nickname。此约束可以通过终端实体页面应用到证书注册和续订。

证书不能具有相同的主题名称，除非一个证书已过期或撤销（而不在冻结时）。因此，活跃证书无法共享主题名称，但例外：如果证书具有不同的密钥使用位，则它们可以共享相同的主题名称，因为它们具有不同的用途。

表 B.34. 唯一主题名称约束配置参数

参数	描述
enableKeyUsageExtensionChecking	可选设置，允许证书具有与密钥使用设置相同的主题名称。这是 true 或 false。默认值为 true，它允许重复主题名称。

B.2.14. 有效约束

Validity 约束检查证书请求中的有效期是否满足标准。

提供的参数必须是可配置的值。例如，提供尚未通过的时间的 notBefore 参数不会被接受，且提供时间早于 notBefore 时间的 notAfter 参数将不接受。

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 时间和 notAfter 时间之间的差别（以天为单位）必须小于范围值，否则此约束将被拒绝。
notBeforeCheck	验证范围不在宽限期内。当 NotBeforeCheck 布尔值参数设为 true 时，系统将检查 notBefore 时间不会大于当前时间，再加上 notBeforeGracePeriod 值。如果 notBeforeTime 不在当前时间和 notBeforeGracePeriod 值之间，这个约束将被拒绝。
notBeforeGracePeriod	未过期时间后宽限期（以秒为单位）。如果 notBeforeTime 不在当前时间和 notBeforeGracePeriod 值之间，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	验证给定时间是否在过期期后没有。当 notAfterCheck 布尔值参数设为 true 时，系统将检查 notAfter 时间不会超过当前时间。如果当前时间超过 notAfter 时间，这个约束将被拒绝。

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，允许向证书添加任意数量的其他字段。证书扩展提供了一种向证书添加信息（如备用主题名称和用量限制）的方法。在 PKIX 第 1 部分标准之前开发的较旧的 Netscape 服务器（如 Red Hat Directory Server 和 Red Hat Certificate System）需要定义特定于 Netscape 的扩展。

以下是包含 X.509 v3 扩展的证书部分的示例。证书系统可以以可读的用户打印格式显示证书，如下所示。如本例所示，证书扩展按顺序显示，每个证书只能有一个特定扩展的实例。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具有版本 0x2 （对应于版本 3）。

例 B.4. Pretty-Print 证书扩展示例

```
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)是标识唯一对象的数字字符串，如证书扩展或公司的证书实践声明。证书系统附带一组特定于扩展的配置集插件模块，可启用 X.509 证书扩展添加到服务器问题的证书中。其中一些扩展包含用于指定 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。这可促进共享网络上的机构之间的互操作性。如果发布将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 委派给区域注册授权。在美国，美国美国国家标准(ANSI)管理此注册。

根据情况，使用注册到另一个机构或无法注册 OID 的 OID 可能会产生法律后果。注册可能会受费用购买。如需更多信息，请联系适当的注册机构。

要为自定义对象定义或分配 OID，知道公司的 arc，它是一个私有企业的 OID。如果公司没有 arc，则需要获取一个。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 证书注释的 Netscape 定义的 OID 是 2.16.840.1.113730.1.13。分配给此扩展的 OID 是分级的，包括前一个 Netscape 公司 arc, 2.16.840.1。OID 定义条目为 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扩展，包括任何可选的限定符，或者它必须拒绝证书。由于所有应用程序都将能够解释以 OID 的形式嵌入的公司的自定义扩展，PKIX 标准建议始终标记为非关键。

本节总结了定义为互联网 X.509 版本 3 标准一部分的扩展类型，并指示 PKIX 工作组推荐哪些类型。

本参考总结了有关每个证书的重要信息。详情请查看来自 ITU 的 X.509 v3 标准，以及 互联网 X.509 公钥基础架构 - 证书和 CRL 配置文件(RFC 339)，请参见 RFC 3280。扩展的描述引用讨论扩展的标准草案的 RFC 和部分号；还提供每个扩展的对象标识符(OID)。

证书中的每个扩展都可以指定为 critical 或 noncritical。证书使用的系统（如 Web 浏览器）必须拒绝证书，如果它遇到不识别的关键扩展；但是，如果无法识别非关键扩展，则可以忽略它。

B.3.1. authorityInfoAccess

授权信息访问扩展指示如何以及访问证书签发者的信息。扩展包含一个 accessMethod 和 accessLocation 字段。accessMethod 由 OID 指定，有关在 accessLocation 中命名的签发者信息的类型和格式。

PKIX Part 1 定义了一个 accessMethod (id-ad-calssuers)，以获取 CA 链中用扩展名比证书签发者更高的证书的 CA 列表。然后，accessLocation 字段通常包含一个 URL，指示用于检索列表的位置和协议(LDAP、HTTP 或 FTP)。

在线证书状态协议(RFC 2560)位于 RFC 2560，定义使用 OCSP 验证证书的 accessMethod (id-ad-ocsp)。然后 accessLocation 字段包含一个 URL，指示用于访问验证证书的 OCSP 响应者的位置和协议。

OID

1.3.6.1.5.5.7.1.1

严重性

这个扩展必须是非关键的。

B.3.2. authorityKeyIdentifier

授权密钥标识符扩展标识与用于签署证书的私钥对应的公钥。当签发者有多个签名密钥（如续订 CA 证书时）时，这个扩展很有用。

扩展由以下一个或多个组成：

- 在 keyIdentifier 字段中设置的显式键标识符
- 在 authorityCertIssuer 字段中设置签发者，并在 authorityCertSerialNumber 字段中设置，用于标识证书

如果存在 keyIdentifier 字段，它将用于选择具有匹配 subjectKeyIdentifier 扩展的证书。如果存在 authorityCertIssuer 和 authorityCertSerialNumber 字段，则会使用它们通过 issuer 和 serialNumber

识别正确的证书。

如果没有这个扩展，则只使用签发者名称来识别签发者证书。

PKIX 第 1 部分要求所有证书的扩展，但自签名的 root CA 证书除外。如果还没有建立密钥标识符，PKIX 建议指定 authorityCertIssuer 和 authorityCertSerialNumber 字段。这些字段允许构建完整的证书链，方法是将签发者证书中的 SubjectName 和 CertificateSerialNumber 字段与主题证书的授权密钥标识符扩展中的 authortiyCertIssuer 和 authorityCertSerialNumber 匹配。

OID

2.5.29.35

严重性

这个扩展始终不是关键的，始终被评估。

B.3.3. basicConstraints

此扩展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。所有 CA 证书都应将 cA 组件设置为 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，其值必须大于目前处理的 CA 证书数量，从最终证书开始并移动链。如果省略 pathLenConstraint，那么链中的所有更高级别的 CA 证书不得在存在扩展时包括此组件。

OID

2.5.29.19

严重性

PKIX 第 1 部分要求此扩展被标记为 critical。无论其重要程度如何，都会评估此扩展。

B.3.4. certificatePoliciesExt

证书策略扩展定义一个或多个策略，各自由 OID 和可选的限定符组成。扩展可以包含到签发者证书实践声明的 URI，也可以嵌入签发者信息，如用户以文本形式通知。此信息可以被启用证书的应用程序使用。

如果存在此扩展，PKIX 第 1 部分建议只使用 OID 识别策略，或者只在需要时使用某些推荐的限定符。

OID

2.5.29.32

严重性

这个扩展可能至关重要或非关键。

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 发布点，则应使用它。

如果扩展包含一组 URI 的 DistributionPointName，则 URI 将假定是指向当前 CRL 的指针，因为指定的撤销原因，并将由命名 cRLIssuer 发出。URI 的预期值是为 Subject Alternative Name 扩展定义的值。如果 distributionPoint 省略原因，则 CRL 必须包括吊销。如果 DistributionPoint 省略了 cRLIssuer，则 CRL 必须由签发证书的 CA 发布。

PKIX 建议 CA 和应用程序支持此扩展。

OID

2.5.29.31

严重性

PKIX 建议此扩展标记为非关键，并且对所有证书都支持。

B.3.6. extKeyUsage

扩展的密钥用法扩展表示可以使用认证公钥的目的。这些目的可能还需要或代替在 Key Usage 扩展中指示的基本目的。

扩展的密钥用法扩展必须在 OCSP 响应者证书中包含 OCSP 签名，除非签署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签署响应者验证证书的 CA 直接发布。

Key Usage、extend Key Usage 和 Basic Constraints 扩展一起用来定义要使用的证书的目的。应用程序可以使用这些扩展来禁止在不当使用上下文中使用证书。

表 B.36 “PKIX 扩展的密钥用法扩展使用” 列出 PKIX 定义用于这个扩展的使用，表 B.37 “私有扩展密钥用法扩展使用” 列表使用 Netscape 私有定义的。

OID

2.5.29.37

严重性

如果这个扩展被标记为关键，则证书必须仅用于指定目的之一。如果没有标记为关键状态，它将被视为可能用于识别密钥但不会将证书的使用限制到指定目的的公告字段。

表 B.36. PKIX 扩展的密钥用法扩展使用

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]

[a] PKIX 第 1 部分没有定义 OCSP 签名，而是在 RFC 2560 中定义，X.509 Internet Public Key Infrastructure Online 证书状态协议 - OCSP。

表 B.37. 私有扩展密钥用法扩展使用

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

严重性

PKIX 第 1 部分建议将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目的。Key Usage、extend Key Usage 和 Basic Constraints 扩展协同工作，以指定可以使用证书的目的。

如果完全包含这个扩展，请设置位，如下所示：

- 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)。
- 用于某些 S/MIME 签名证书和对象签名证书的 非 验证(1)。

WARNING

使用这个位是相似的。在为任何证书设置前，请仔细考虑其使用的法律后果。

- 用于 SSL 服务器证书和 S/MIME 加密密钥(2)。
- 当主题的公钥用于加密用户数据而不是密钥材料时，Data Encipherment (3)。
- 当主题的公钥用于 密钥协议时，KeyAgreement (4)。
- KeyCertSign (5)用于所有 CA 签名证书。
- 用于签署 CRL 的 CA 签名证书的 cRLSign (6)。
- encipherOnly (7)，如果公钥仅用于加密数据。如果设置了此位，则还应设置 keyAgreement。
- 如果公钥仅用于解码数据，则 decipherOnly (8)。如果设置了此位，则还应设置 keyAgreement。

表 B.38 “证书使用和清理密钥用法表” 总结了典型证书使用的指南。

如果存在 keyUsage 扩展并标记为 critical，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；如果扩展不存在或不存在，则允许所有类型的用法。

如果存在 keyUsage 扩展，critical 或 not，它将用于从给定操作的多个证书中选择。例如，它用于区分具有单独证书和密钥对操作的用户的单独签名和加密密钥。

OID

2.5.29.15

严重性

这个扩展可能至关重要或非关键。PKIX 第 1 部分建议在使用时将其标记为 critical。

表 B.38. 证书使用和清理密钥用法表

证书的目的	所需的密钥用法位
CA 签名	• keyCertSign • cRLSign
SSL 客户端	digitalSignature
SSL 服务器	keyEncipherment
S/MIME 签名	digitalSignature
S/MIME 加密	keyEncipherment
证书签名请求	keyCertSign
对象签名	digitalSignature

B.3.9. nameConstraints

此扩展只在 CA 证书中使用，定义认证路径中后续证书中的所有主题名称都必须位于的名字空间。

OID

2.5.29.30

严重性

PKIX 第 1 部分要求此扩展被标记为 critical。

B.3.10. OCSPNocheck

扩展旨在包含在 OCSP 签名证书中。扩展告知 OCSP 客户端可以在不查询 OCSP 响应程序的情况下信任签名证书（因为回复可能会被 OCSP 响应程序再次签名，客户端也会再次请求签名证书的有效性状态）。这个扩展为 null-valued，其含义由其存在或不存在。

由于证书中存在这个扩展将导致 OCSP 客户端信任使用该证书签名的证书，因此应仔细管理此扩展。如果 OCSP 签名密钥被破坏，那么 PKI 中验证证书的整个过程会在证书的有效期间破坏。因此，使用 OCSPNocheck 的证书应该在短生命周期内发布，并经常续订。

OID

1.3.6.1.5.5.7.48.4

严重性

这个扩展应该不是关键的。

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 要求如果存在，此扩展不得由空序列组成。必须至少有两个可用字段之一。

OID

2.5.29.36

严重性

这个扩展可能至关重要或非关键。

B.3.12. policyMappings

策略映射扩展仅在 CA 证书中使用。它列出了一个或多个 OID 对，用于表示一个 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可能很有用。

CA 和应用程序可以支持此扩展。

OID

2.5.29.33

严重性

这个扩展必须是非关键的。

B.3.13. privateKeyUsagePeriod

Private Key Usage Period 扩展允许证书签发者为私钥指定与证书本身不同的有效期周期。此扩展用于数字签名密钥。

注意

PKIX 第 1 部分建议使用此扩展。符合 PKIX 部分 1 的 CA 不能 生成具有此扩展的证书。

OID

2.5.29.16

B.3.14. subjectAltName

主题备用名称扩展包括 CA 绑定到认证公钥绑定的一个或多个替代（非 X.500）名称。除了证书的主题名称或替换证书外，还可使用它。定义的名称表单包括互联网电子邮件地址(SMTP，如 RFC-822 中定义的、DNS 名称、IP 地址（IPv4 和 IPv6)以及统一资源标识符(URI)。

PKIX 需要此扩展，用于由主题字段中使用的 X.500 可分辨名称(DN)标识的实体。PKIX Part 1 描述了此扩展和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、证书主题名称字段或两者中提供。如果电子邮件地址是主题名称的一部分，它必须采用 PKCS1149 定义的 EmailAddress 属性的形式。支持 S/MIME 的软件必须能够从 Subject Alternative Name 扩展或从主题名称字段读取电子邮件地址。

OID

2.5.29.17

严重性

如果证书的 subject 字段为空，这个扩展必须标记为 critical。

B.3.15. subjectDirectoryAttributes

Subject Directory Attributes 扩展处理证书的主题所需的目录属性值。不建议将其作为提议的 PKIX 标准的重要部分，但可能在本地环境中使用。

OID

2.5.29.9

严重性

PKIX 第 1 部分要求此扩展被标记为非关键。

B.3.16. subjectKeyIdentifier

Subject Key Identifier 扩展标识此证书认证的公钥。如果给定主题名称有多个可用的公钥，这个扩展提供了一种区分公钥的方法。

此扩展的值应该通过执行证书的 DER 编码 subjectPublicKey 的 SHA-1 哈希来计算，如 PKIX 推荐的。Subject Key Identifier 扩展与 CA 证书的授权密钥标识符扩展一起使用。如果 CA 证书具有 Subject Key Identifier 扩展，则验证的证书授权密钥标识符扩展的密钥标识符应与 CA 主题密钥标识符扩展的密钥标识符匹配。在这种情况下，verifier 不需要重新计算密钥标识符。

PKIX 第 1 部分需要对所有 CA 证书进行这个扩展，并推荐使用所有其他证书。

OID

2.5.29.14

严重性

这个扩展始终不是关键的。

B.4. CRL 扩展

B.4.1. 关于 CRL 扩展

自初始发布以来，对 CRL 格式的 X.509 标准已被修改，以在 CRL 中包含其他信息。此信息通过 CRL 扩展添加。

ANSI X9 和 ISO/IEC/ITU 为 X.509 CRLs [X.509] [X9.55] 定义的扩展允许额外的属性与 CRL 关联。RFC 5280 提供了 互联网 X.509 公钥基础架构证书和 CRL 配置文件，建议在 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，并针对组织或业务执行信息。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

抽象语法 Notation One (ASN.1)和 Distinguished Encoding Rules (DER)标准在 CCITT Recommendations X.208 和 X.209 中指定。有关 ASN.1 和 DER 的快速摘要，请参阅 Lay man's Guide to a Subset of ASN.1, BER, 和 DER，该指南位于 RSA theoratories' http://www.rsa.com。

B.4.1.1. CRL 扩展的结构

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 字段中值的 ASN.1 类型以及如何解释值。当一个扩展出现在 CRL 中时，OID 显示为扩展 ID 字段(extnID)，相应的 ASN.1 编码结构显示为 octet 字符串的值(extnValue); 示例显示在 例 B.4 “Pretty-Print 证书扩展示例” 中。
- 名为 critical 的标志或布尔值字段。

  分配给此字段的 true 或 false 值指示扩展对 CRL 是关键还是不重要。

- 如果扩展至关重要，并且 CRL 发送到不根据扩展的 ID 了解扩展的应用程序，应用程序必须拒绝 CRL。
- 如果扩展不重要，并且 CRL 发送到不根据扩展 ID 了解扩展的应用程序，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 octet 字符串。

接收 CRL 的应用会检查扩展 ID，以确定它是否可以识别 ID。如果它可以，它可以使用扩展 ID 来决定所使用的值类型。

B.4.1.2. CRL 和 CRL 条目扩展示例

以下是 X.509 CRL 版本 2 扩展的示例。证书系统可以以可读的用户打印格式显示 CRL，如下所示。如示例所示，CRL 扩展按顺序出现，每个 CRL 只能有一个特定扩展的实例。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，仅包含自上次 CRL 发布以来的更改。任何包含 delta CRL 指示符扩展的 CRL 都是一个 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 建议的标准还定义了 CRL 的扩展，它提供了将额外属性与互联网 CRL 关联的方法。以下是两种类型之一：对 CRL 本身的扩展，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 条目扩展”

B.4.2.1. CRL 的扩展

以下 CRL 描述定义为互联网 X.509 v3 公钥基础架构建议的标准的一部分。

- 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。newestCRL 扩展放在完整的 CRL 中，以指示在哪里查找最新的 delta CRL。

OID

1.3.6.1.5.5.7.1.1

严重性

PKIX 要求此扩展不能至关重要。

参数

表 B.39. 授权信息访问配置参数

参数	描述
enable	指定规则是否启用或禁用。默认值为禁用此扩展。
critical	设置扩展是否标记为关键；默认值为非关键。

参数	描述
numberOfAccessDescriptions	表示访问描述的数量，从 0 到任意正整数；默认值为 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 以关闭窗口。重新打开规则的编辑窗口，以及设置点的字段将存在。
accessMethodn	此参数唯一接受的值是 calssuers。当可用信息列表可用于验证 CRL 上的签名的证书时，会使用 calssuers 方法。当 CRL 中包含 AIA 扩展时，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 设置为 DirectoryName，则该值必须是字符串，格式为 X.500 名称，类似于证书中的主题名称。例如：CN=CACentral,OU=Research Dept,O=Example company,C=US。 如果 accessLocationType 设为 URI，则名称必须是 URI；URI 必须是绝对路径，且必须指定主机。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” 中的证书扩展中的讨论。

PKIX 标准建议 CA 必须在所有 CRL 中包括此扩展，因为 CA 的公钥可能会改变，例如，当密钥被更新时，或者 CA 可能存在多个签名密钥，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。在证书上验证签名时，其他应用程序需要知道在签名中使用哪个密钥。

OID

2.5.29.35

参数

表 B.40. AuthorityKeyIdentifierExt 配置参数

参数	描述
enable	指定规则是否启用或禁用。默认值为禁用此扩展。
critical	设置扩展是否标记为关键；默认值为非关键。

B.4.2.1.3. CRLNumber

CRLNumber 扩展指定 CA 发布的每个 CRL 的连续数字。它允许用户轻松确定何时将特定的 CRL 取代另一个 CRL。PKIX 要求所有 CRL 都具有此扩展。

OID

2.5.29.20

严重性

这个扩展不能至关重要。

参数

表 B.41. CRLNumber 配置参数

参数	描述
enable	指定是否启用了规则，这是默认设置。
critical	设置扩展是否标记为关键；默认值为非关键。

B.4.2.1.4. deltaCRLIndicator

deltaCRLIndicator 扩展生成 delta CRL，仅包含从最后一个 CRL 吊销的证书列表，还包括对基础 CRL 的引用。这会更新本地数据库，同时忽略已存在于本地数据库中的未更改的信息。这可以显著提高以 CRL 结构以外的格式存储撤销信息的应用程序处理时间。

OID

2.5.29.27

严重性

PKIX 要求如果扩展存在，则此扩展至关重要。

参数

表 B.42. DeltaCRL 配置参数

参数	描述
enable	设置是否启用了规则。默认情况下，它被禁用。
critical	设置扩展是否为关键还是非关键。默认情况下，这很重要。

B.4.2.1.5. FreshestCRL

newestCRL 扩展标识如何获取 delta CRL 信息。newestCRL 扩展放在完整的 CRL 中，以指示在哪里查找最新的 delta CRL。

OID

2.5.29.46

严重性

PKIX 要求此扩展必须为非关键。

参数

表 B.43. FreshestCRL 配置参数

参数	描述
enable	*设置是否启用了扩展规则。默认情况下禁用此设置。*
critical	*将扩展标记为 critical 或 noncritical。默认值为 noncritical。*
numPoints	*表示 delta CRL 的发布点数量，从 0 到任意正整数；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 以关闭窗口。重新打开规则的编辑窗口，以及设置这些点的字段将存在。*
pointTypen	*指定 n emit 点的发布点的类型。对于 numPoints 中指定的每个数字，有相等的 pointType 参数数。选项可以是 DirectoryName 或 URIName。*
pointNamen	*如果将 pointType 设置为 directoryName，则该值必须是 X.500 名称形式的字符串，类似于证书中的主题名称。例如：CN=CACentral,OU=Research Dept,O=Example company,C=US。* *如果 pointType 设为 URIName，则名称必须是 URI；URI 必须是绝对路径，且必须指定主机。例如：http://testCA.example.com/get/crls/here/。*

B.4.2.1.6. issuerAltName

Issuer Alternative Name 扩展允许额外的身份与 CRL 的签发者关联，如绑定属性，如电子邮件地址、DNS 名称、IP 地址(IPv4 和 IPv6)，以及一个统一的资源指示器(URI)，以及 CRL 签发者。详情请查看 第 B.3.7 节 “issuerAltName 扩展” 中的证书扩展中的讨论。

OID

2.5.29.18

参数

表 B.44. IssuerAlternativeName 配置参数

参数	描述
enable	设置是否启用了扩展规则；默认情况下，这是禁用的。
critical	设置扩展是否至关重要；默认情况下，这并不重要。
numNames	设置扩展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，它们必须具有适当的值，否则规则会返回错误。通过更改此字段中指定的值来更改身份总数；扩展中包含的身份总数没有限制。每个配置参数集合都通过从此字段值派生的整数来区分。例如，如果 numNames 参数设置为 2，则派生的整数为 0 和 1。
nameTypen	指定常规名称类型；可以是以下任意一种： • 如果 名称是互联网电子邮件地址，rr822Name。 • 如果 名称是 X.500 目录名称，

参数	描述
	则 directoryname。 • 如果名称是 DNS 名称，则 dnsName。 • ediPartyName 如果名称是 EDI 方名称。 • 如果名称是 URI （默认）的 URL。 • 如果名称是 IP 地址，则 ipaddress。IPv4 地址的格式必须是 n.n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 如果名称是对象标识符，则 OID。 • otherName 如果名称采用任何其他名称形式；这支持 PrintableString、IA5String、UUI8String、BMPString 、任何、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。

参数	描述
	• 对于 rfc822Name，该值必须是有效的互联网电子邮件地址，格式为 local-part@domain。 • 对于 directoryName，该值必须是字符串 X.500 名称，类似于证书中的主题名称。例如：CN=CACentral,OU=Research Dept,O=Example company,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，公司示例。 • 对于 URL，该值必须是非相对 URI。例如：http://testCA.example.com。 • 对于 iPAddress，该值必须是以点分开的数字组件表示法指定的有效 IP 地址。它可以是 IP 地址或 IP 地址，包括子网掩码。IPv4 地址的格式必须是 n. n.n.n 或 n .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，其 IPv6 地址用冒号和以句点分开的子网掩码分开。例如，0:0:0:0:0:0:13.1.68.3,FF01::43,0:0:0:0:0:0:13.1.68.3,FF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0 , 和 FF01::43,FFFF:FFFF:FF:FF:FF:FF:FF:FFFF0000。 • 对于 OID，该值必须是以点分隔的数字组件表示法中指定的唯一有效 OID。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

参数	描述
	生产环境中，遵守 ISO 规则来定义 OID 和 ID 的注册子树。 • 对于 otherName，名称可以是任何其他格式；这支持 PrintableString、IA5String、UUI8String、BMPString 、任何、和 KerberosName。PrintableString,IA5String,UTF8String,BMPString, 和 any 将字符串设置为指定子树的 base-64 编码文件，如 /var/lib/pki/pki-tomcat/ca/othername.txt。KerberosName 格式为 Realm|NameType|NameStrings，如 realm1|0|userID1,userID2。名称必须是文件的绝对路径，该文件包含其 base-64 编码格式的通用名称。例如：/var/lib/pki/pki-tomcat/ca/extn/ian/othername.txt。

B.4.2.1.7. issuingDistributionPoint

颁发发布点 CRL 扩展标识特定 CRL 的 CRL 发布点，并指示它涵盖的撤销类型，如只撤销最终证书、仅 CA 证书或撤销具有有限原因代码的证书。

PKIX Part 我不需要此扩展。

OID

2.5.29.28

严重性

PKIX 要求如果扩展存在，则此扩展至关重要。

参数

表 B.45. IssuingDistributionPoint 配置参数

参数	描述
enable	设置是否启用了扩展；默认禁用。
critical	将扩展标记为 critical、default 或 noncritical。
pointType	指定来自以下内容的发布发布点的类型： • directoryName 指定类型是 X.500 目录名称。 • URI 指定类型是一个统一的资源类型。 • RelativeToIssuer 指定类型是一个相对可分辨名称(RDN)，它代表 DN 的单个节点，如 ou=Engineering。

参数	描述
pointName	提供发布点的名称。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，名称必须是一个 URI，它是一个绝对路径并指定主机。例如：http://testCA.example.com/get/crls/here/。  注意 CRL 可以存储在与 CRL 发布点对应的目录中，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原因代码。 感知值是原因代码(未指定、键组合、cACompromise、affiliationChanged、替换的、cessationOfOperation、certHold 和删除FromCRL)的组合。如果发行版点包含带有所有原因代码（默认）吊销的证书，请将该字段留空。
onlyContainsCACerts	指定仅在设置时包含用户证书。默认情况下不设置，这意味着发行版点包含所有类型的证书。
indirectCRL	指定发行版点包含间接 CRL；默认情况下，不会选择。

B.4.2.2. CRL 条目扩展

以下章节列出了定义为互联网 X.509 v3 公钥基础架构建议标准一部分的 CRL 条目扩展类型。所有这些扩展都不是关键的。

B.4.2.2.1. certificateIssuer

证书签发者扩展标识与间接 CRL 中条目关联的证书签发者。

此扩展仅用于间接 CRL，不受证书系统支持。

OID

2.5.29.29

B.4.2.2.2. invalidityDate

Invalidity Date 扩展提供了私钥被泄露或证书无效的日期。

OID

2.5.29.24

参数

表 B.46. InvalidityDate 配置参数

参数	*描述*
enable	*设置扩展规则是否启用或禁用。默认情况下启用此设置。*
critical	*将扩展标记为 critical 或 noncritical；默认情况下，这并不重要。*

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	*设置扩展规则是否启用或禁用。默认情况下启用此设置。*
critical	*将扩展标记为 critical 或 noncritical。默认情况下，这不是不关键的。*

B.4.3. Netscape-Defined 证书扩展参考

Netscape 为其产品定义了某些证书扩展。有些扩展现已过时，其他扩展已被 X.509 建议的标准中定义的扩展替代。所有 Netscape 扩展都应标记为非关键扩展，因此其证书中存在不会使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”。

如果证书中存在扩展，它会将证书限制为其中指定的使用。如果没有扩展名，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，其中单独的位位置（在设置时）认证特定使用的证书，如下所示：

- 位 0：SSL 客户端证书
- 位 1：SSL 服务器证书
- 第 2 位：S/MIME 证书
- 第 3 位：对象签名证书
- 第 4 位： reserved
- 第 5 位：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

此扩展名的值是 IA5String。这是一个注释，可在查看证书时向用户显示。

OID

2.16.840.1.113730.13

附录 C. 发布模块参考

默认通过证书管理器配置几个发布程序、映射程序和规则模块。

- 第 C.1 节 “发布程序插件模块”
- 第 C.2 节 “映射程序插件模块 ”
- 第 C.3 节 “规则实例”

C.1. 发布程序插件模块

本节论述了为证书管理器提供的发布程序模块。证书管理器使用该模块启用和配置特定的发布程序实例。

- 第 C.1.1 节 “FileBasedPublisher”
- 第 C.1.2 节 “LdapCaCertPublisher”
- 第 C.1.3 节 “LdapUserCertPublisher”
- 第 C.1.4 节 “LdapCrlPublisher”
- 第 C.1.5 节 “LdapDeltaCrlPublisher”
- 第 C.1.6 节 “LdapCertificatePairPublisher”
- 第 C.1.7 节 “OCSPPublisher”

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。此插件可根据配置发布程序时选择的复选框发布 base-64 编码文件、DER 编码文件或两者。证书和 CRL 内容可以通过使用 PrettyPrintCert 和 PrettyPrintCRL 工具转换文件来查看。有关在 base-64 和 DER 编码证书和 CRL 中查看内容的详情，请参考 第 9.11 节 “查看证书和 CRL 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
发布者 ID	为 publisher 指定名称，一个没有空格的字母字符串。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，也可以相对于证书系统实例目录。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发布 CA 证书到 CA 的目录条目的 caCertificate;binary 属性。

模块将 CA 条目的对象类转换为 pkiCA 或 certifiedAuthority （如果还没有使用）。同样，如果 CA 没有其他证书，它还会在未发布时删除 pkiCA 或 certified Authority 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，以将 CA 证书发布到该目录。

表 C.2. LdapCaCertPublisher 配置参数

参数	描述
caCertAttr	指定发布 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发布用户证书到用户的目录条目的 userCertificate;binary 属性。

此模块用于将任何最终用户证书发布到 LDAP 目录。最终证书的类型包括 SSL 客户端、S/MIME、SSL 服务器和 OCSP 响应程序。

在安装过程中，证书管理器会自动创建 LdapUserCertPublisher 模块实例，以将最终用户证书发布到该目录。

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 directory 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发布 CRL 到目录条目的 certificateRevocationList;binary 属性。

在安装过程中，证书管理器会自动创建 LdapCrlPublisher 模块实例，用于将 CRL 发布到 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 directory 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发布 deltaRevocationList 属性到目录条目的 deltaRevocationList 属性。

在安装过程中，证书管理器会自动创建 LdapDeltaCrlPublisher 模块实例，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 directory 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发布 CA 目录条目的 跨 CertPair;binary 属性。

模块还会将 CA 条目的对象类转换为 pkiCA 或 CertificationAuthority （如果还没有使用）。同样，如果 CA 没有其他证书，它还会在未发布时删除 pkiCA 或 certified Authority 对象类。

在安装过程中，证书管理器会自动创建一个名为 LdapCertificatePairPublisher 模块的实例，名为 LdapCrossCertPairPublisher，以将跨签名的证书发布到该目录。

表 C.6. LdapCertificatePairPublisher Parameters

参数	描述
crossCertPairAttr	指定发布 CA 证书的 LDAP 目录属性。这必须是 **crossCertificatePair;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 插件模块配置证书管理器，将其 CRL 发布到在线证书状态管理器。

证书管理器在安装时不会创建 OCSPPublisher 模块的任何实例。

表 C.7. OCSPPublisher 参数

参数	描述
host	指定在线证书状态管理器的完全限定主机名。
端口	指定在线证书状态管理器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	设置是否使用客户端（基于证书）验证来访问 OCSP 服务。
nickname	在 OCSP 服务的数据库中提供证书的别名，以用于客户端身份验证。这仅在 **enableClientAuth** 选项设置为 true 时使用。

C.2. 映射程序插件模块

本节论述了为证书管理器提供的映射程序插件模块。这些模块配置证书管理器以启用和配置特定的映射程序实例。

可用的映射程序插件模块包括：

- *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 插件模块将证书管理器配置为自动在 LDAP 目录中为 CA 创建条目，然后通过从证书请求、证书主题名称、证书扩展、证书扩展和属性变量断言(AVA)来声明条目的 DN 来将 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。

如果发布目录中已存在 CA 条目，并且分配给该映射器的 dnPattern 参数的值已更改，但 uid 和 o 属性相同，则映射器无法创建第二个 CA 条目。例如，如果目录已经具有 uid=CA,ou=Marketing,o=example.com 的 CA 条目，并且将映射器配置为创建另一个带有 uid=CA,ou=Engineering,o=example.com 的 CA 条目，则操作会失败。

操作可能会失败，因为目录会将 UID 唯一 插件设置为特定的基本 DN。此设置可防止目录在该基本 DN 下有两个带有相同 UID 的条目。在本例中，它可防止目录在 o=example.com 下有两个条目，它们具有相同的 UID CA。

如果映射器无法创建第二个 CA 条目，请检查设置 UID 唯一插件的基本 DN，并检查目录中是否已存在具有相同 UID 的条目。如有必要，调整 mapper 设置，删除旧的 CA 条目，注释掉插件，或者手动创建条目。

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。映射程序命名如下：

- *用于 CRL 的 LdapCrlMap （请参阅 第 C.2.1.2 节 “LdapCrlMap”）*
- *CA 证书的 LdapCaCertMap （请参阅 第 C.2.1.1 节 “LdapCaCertMap”）。*

表 C.8. LdapCaSimpleMap 配置参数

参数	描述
createCAEntry	如果选中（默认）创建 CA 条目。 如果选中，证书管理器首先会尝试在目录中为 CA 创建条目。如果证书管理器在创建条目中成功，它会尝试将 CA 的证书发布到该条目。如果没有选择此项，则必须已存在该条目才能发布到该条目。

参数	描述
dnPattern	指定证书管理器应该用来在发布目录中搜索 CA 条目的 DN 模式。**dnPattern** 的值可以是用逗号分开的 AVAs 列表。AVA 可以是变量，如 **cn=$subj.cn**，证书管理器可以从证书主题名称或常量生成，如 **o=Example 公司**。 如果 CA 证书在其主题名称中没有 **cn** 组件，请调整 CA 证书映射 DN 模式，以反映要发布 CA 证书的目录中的 DN。例如，如果 CA 证书主题 DN 是 **o=Example 公司**，目录中的 CA 条目是 **cn=Certificate Authority, o=Example 公司**，则模式为 **cn=Certificate Authority, o=$subj.o**。 ● 示例 1： **uid=CertMgr, o=Example** ● 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** 在上例中，**$req** 从证书请求中获取属性，**$subj** 从证书主题名称获取属性，**$ext** 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。

此映射器在目录中为 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。

此映射器在目录中为 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 插件模块将证书管理器配置为通过搜索与证书主题名称匹配的 LDAP 条目 DN 来将证书映射到 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，如果证书主题名称为 uid=jdoe、o=Example 公司、c=US，在搜索条目的目录时，证书管理器仅搜索 DN uid=jdoe、o=Example 公司、c=US 的条目。

如果没有找到匹配的条目，服务器会返回错误，且不会发布证书。

此映射程序不需要任何参数的任何值，因为它从证书获取所有值。

C.2.3. LdapSimpleMap

LdapSimpleMap 插件模块将证书管理器配置为通过从证书请求中指定的组件派生条目、证书名称、证书扩展和属性变量断言(AVA)来将证书映射到 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程序的映射程序规则。在安装过程中，证书管理器会自动创建一个简单映射程序模块实例，名为 LdapUserCertMap。默认映射程序将各种类型的最终用户证书映射到对应的目录条目。

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，也可以是常量，如 o=Example 公司。

- 示例 1： uid=CertMgr, o=Example
- 示例 2: cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US
- Example 3: uid=$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou

在示例中，$req 从证书请求中获取属性，$subj 从证书主题名称获取属性，$ext 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，以使用可配置的 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。

此映射器需要主题 DN 的确切模式，因为证书管理器在目录中搜索与整个主题 DN 完全匹配的值的属性。例如，如果指定的 LDAP 属性是 certSubjectDN，证书主题名称为 uid=jdoe、o=Example 公司、c=US，则证书管理器搜索具有属性 certSubjectDN=uid=jdoe、o=Example 公司、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中。

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，但这可以配置为任何 LDAP 属性。
searchBase	指定启动属性搜索的基本 DN。允许的值是 LDAP 条目的有效 DN，如 o=example.com、c=US。

C.2.5. LdapDNCompsMap

LdapDNCompsMap 插件模块实现 DN 组件映射器。此映射程序通过从组件构建条目的 DN 来将证书映射到 LDAP 目录条目，如 cn、ou、o 和 c，然后在证书主题名称中指定，然后使用它作为搜索 DN 在目录中查找条目。映射程序找到以下条目：

- 用于发布 CA 证书和 CRL 的目录中的 CA 条目。
- 用于发布最终用户证书的目录中的最终用户条目。

映射程序采用 DN 组件来构建搜索 DN。映射程序也采用可选的 root 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，以组成子树的搜索过滤器。如果没有配置 DN 组件，服务器将使用基本 DN 进行子树。如果基础 DN 为 null 且没有 DN 组件匹配，则返回错误。如果没有 DN 组件和过滤组件匹配，则返回错误。如果过滤器组件为 null，则会执行基础搜索。

DNComps 和 filterComps 参数接受有效的 DN 组件或用逗号分隔的属性。参数不接受属性的多个条目；例如，filterComps 可以设置为 cn,ou 2,ou1。要使用相同属性的多个实例创建过滤器，如如果目录条目包含多个 ou s，请修改 LdapDNCompsMap 模块的源代码。

以下组件通常在 DN 中使用：

- 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 代表目录中的组织单元。
- o 代表目录中的一个机构。
- l 代表本地性（城市）。
- st 代表状态。
- c 代表国家/地区。

例如，以下 DN 代表名为 Jane Doe 的用户，其适用于示例公司（位于 美国 加利州）的销售部门：

```
cn=Jane Doe, ou=Sales, o=Example Corporation, l=Mountain View, st=California, c=US
```

证书管理器可以使用其中一些或全部组件(cn、ou、o、l、st、c)来构建用于搜索目录的 DN。在创建映射程序规则时，可以为服务器指定用于构建 DN 的服务器；即，与目录中属性匹配的组件。这通过 dnComps 参数设置。

例如，组件 cn、ou、o 和 c 设置为 dnComps 参数的值。要在目录中找到 Jane Doe 条目，证书管理器通过从证书读取 DN 属性值来构建以下 DN，并使用 DN 作为搜索目录的基础：

```
cn=Jane Doe, ou=Sales, o=Example Corporation, c=US
```

- 主题名称不需要在 dnComps 参数指定所有组件。服务器忽略不属于主题名称的任何组件，如本例中的 l 和 st。

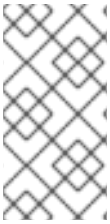
- 未指定组件不用于构建 DN。在示例中，如果没有包含 ou 组件，服务器使用这个 DN 作为搜索目录的基础：

  ```
  cn=Jane Doe, o=Example Corporation, c=US
  ```

对于 dnComps 参数，请输入这些 DN 组件，证书管理器可用于精确组成 LDAP DN。然而，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，证书管理器可能没有获得与 DN 不同的单个匹配条目。例如，主题名称 cn=Jane Doe, ou=sales, o=Example 公司, c=US 可能与两个名为 Jane Doe 的用户匹配。如果发生了这种情况，证书管理器需要额外的条件来确定哪个条目与证书的主题对应。

要指定证书管理器必须用来区分目录中的不同条目的组件，请使用 filterComps 参数；详情请参阅 表 C.10 “LdapDNCompsMap 配置参数”。例如，如果 cn、ou、o 和 c 是 dnComps 参数的值，则仅在 l 属性可用于区分具有相同 cn、ou、o 和 c 值的条目时，才会为 filterComps 参数输入 l。

如果两个 Jane Doe 条目由 uid 属性的值区分 - 一个条目的 uid 是 janedoe1，其他条目的 uid 为 janedoe2 - 证书的主题名称可以设置为包含 uid 组件。



注意

e、l 和 st 组件不包含在为结束实体提供的标准证书请求表单中。这些组件可以添加到表单中，或者在以证书颁发形式编辑主题名称时插入这些组件。

C.2.5.1. LdapDNCompsMap 的配置参数

使用这个配置，证书管理器通过使用 dnComps 值形成 DN 和 filterComps 值，将证书与 LDAP 目录中的证书映射，以形成子树的搜索过滤器。

-

如果 formed DN 为 null，服务器将使用 baseDN 值作为子树。如果被组成的 DN 和基本 DN 都为空，服务器会记录错误。

- *如果过滤器为 null，服务器将使用 baseDN 值进行搜索。如果过滤器和基本 DN 都为空，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，服务器将使用基本 DN 值在目录中启动其搜索。
dnComps	指定证书管理器应开始搜索与 CA 或结束实体信息匹配的 LDAP 条目的位置。 例如，如果 **dnComps** 使用 DN 的 **o** 和 **c** 属性，服务器会从目录中的 **o=***org*,**c=***country* 条目启动搜索，其中 *org* 和 *country* 将被替换为证书中 DN 的值。 如果 **dnComps** 字段为空，服务器将检查 **baseDN** 字段，并在 DN 中搜索与 **filterComps** 参数值指定的过滤器匹配的目录树。 允许的值是有效的 DN 组件或用逗号分开的属性。
filterComps	指定证书管理器应用于过滤搜索结果中的条目的组件。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，搜索可以成功，并且服务器可以选择执行验证。例如，如果 **filterComps** 设为使用电子邮件和用户 ID 属性(**filterComps=e,uid**)，服务器将搜索目录，其值为电子邮件和用户 ID 的值与从证书收集的信息匹配。 允许的值是以逗号分开的证书 DN 中的有效目录属性。过滤器的属性名称需要来自证书的属性名称，而不是来自 LDAP 目录中的属性名称。例如，大多数证书都有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 可用于发布 CA 证书到 LDAP 目录。

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的类型。
predicate		为发布者指定一个 predicate。
enable	**是**	启用规则。
mapper	**LdapCaCertMap**	指定与规则一起使用的映射程序。有关映射程序的详情，请查看第 C.2.1.1 节 “LdapCaCertMap”。
publisher	**LdapCaCertPublisher**	指定与规则一起使用的发布程序。有关发布程序的详情，请查看第 C.1.2 节 “LdapCaCertPublisher”。

C.3.2. LdapXCertRule

LdapXCertRule 用于发布跨对证书到 LDAP 目录。

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的类型。
predicate		为发布者指定一个 predicate。
enable	**是**	启用规则。
mapper	**LdapCaCertMap**	指定与规则一起使用的映射程序。有关映射程序的详情，请查看第 C.2.1.1 节 “LdapCaCertMap”。
publisher	**LdapCrossCertPairPublisher**	指定与规则一起使用的发布程序。有关这个发布程序的详情，请查看第 C.1.6 节 “LdapCertificatePairPublisher”。

C.3.3. LdapUserCertRule

LdapUserCertRule 用于将用户证书发布到 LDAP 目录。

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的类型。
predicate		为发布者指定一个 predicate。
enable	**是**	启用规则。
mapper	**LdapUserCertMap**	指定与规则一起使用的映射程序。有关映射程序的详情，请查看第 C.2.3 节 “LdapSimpleMap”。
publisher	**LdapUserCertPublisher**	指定与规则一起使用的发布程序。有关发布程序的详情，请查看第 C.1.3 节 “LdapUserCertPublisher”。

C.3.4. LdapCRLRule

LdapCRLRule 用于将 CRL 发布到 LDAP 目录。

表 C.14. LdapCRL 规则配置参数

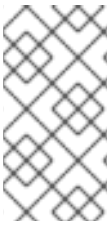
参数	值	描述
type	**crl**	指定将要发布的证书的类型。
predicate		为发布者指定一个 predicate。
enable	**是**	启用规则。
mapper	**LdapCrlMap**	指定与规则一起使用的映射程序。有关映射程序的详情，请查看第 C.2.1.2 节 “LdapCrlMap”。
publisher	**LdapCrlPublisher**	指定与规则一起使用的发布程序。有关发布程序的详情，请查看第 C.1.4 节 “LdapCrlPublisher”。

附录 D. ACL 参考

本节介绍了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 资源提供默认的 ACl。每个子系统仅包含与那个子系统相关的 ACL。

D.1. 关于 ACL 配置文件

访问控制 是设置可以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 目录服务并使用 Java 控制台 - CA、KRA、OCSP 和 TKS - 所有子系统都实施 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

本节仅提供有关访问控制概念的简要概述。红帽目录服务器 管理指南中的管理访问控制 一章中更详细地描述了访问控制。

证书系统 ACL 文件是由内部数据库加载的 LDIF 文件。单个 ACL 定义为 resourceACLS 属性，它标识受保护的子系统区域，然后定义一个要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
```

允许或拒绝对资源的访问的每个规则称为访问控制指令(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系统子系统使用的特定插件类。这会将每个 ACL 专注于子系统执行的特定功能，从而为实例提供更高的安全性，并对应用 ACL 更好地控制。

例 D.1. 列出证书配置文件的默认 ACL

```
resourceACLS: certServer.ca.profiles:list:allow (list) group="Certificate Manager
Agents":Certificate Manager agents may list profiles
```

由于每个子系统(CA、KRA、GADP 和 TKS)都有自己的资源用于其操作，因此每个子系统实例都有自己的 acl.ldif 文件及其自己的定义的 ACL。

每个 ACI 定义了可以执行的操作或行为(右侧)，以及 ACI 应用到什么(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，对目录条目的权限列表相对有限，如搜索、读取、写入和删除。证书系统使用覆盖常见 PKI 任务的额外权限，如撤销、提交和分配。

如果 ACI 中没有显式允许某个操作，则会隐式拒绝操作。如果一个 ACI 中明确拒绝某个操作，它会解析显式允许的任何 ACI。始终希望使用拒绝规则来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。这使用几个常见条件（通常为 user= 或 group= ）进行设置，但还有其他选项，如 ipaddress=，它们定义基于客户端的访问权限，而不是基于条目的访问。如果有多个条件，可以使用双管道(||)运算符组成条件，表示逻辑分布("或")，以及 double ampersand (&&)运算符，表示逻辑使用("and")。例如，group="group1" || "group2"。

resourceACLS 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	ACI 应用到的插件类。
所有操作	ACI 定义中涵盖的每个操作列表。单个 ACI 中可以有多个操作，单个 ***resourceACLS*** 属性中的多个 ACI。
allow|deny	是否允许对目标用户或组执行操作，还是拒绝目标用户或组。
(*操作*)	允许或拒绝的操作。
type=target	用于标识应用到哪些目标。这通常是一个用户（如 **user=***"name"*）或组(**group=***"group"*)。如果有多个条件，可以使用双管道(||)运算符（逻辑 "or"）和 double ampersand (&&)运算符（逻辑"and"）来组成条件。例如，**group="group1" || "group2"**。
description	有关 ACL 的作用的描述。

D.2. 常见 ACL

本节介绍所有四个子系统类型常见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，如日志记录和添加用户和组。

重要

这些 ACL 通常在每个子系统实例的 acl.ldif 文件中发生相同的 ACL。这些不是 共享的 ACL，因此配置文件或设置都由所有子系统实例保存在通用的。与所有其他实例配置一样，这些 ACL 与其他子系统实例独立维护，位于特定于实例的 acl.ldif 文件中。

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"
```

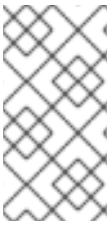
表 D.2. certServer.acl.configuration ACL Summary

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 列表等效器和 ACL 等效器类型。	Allow	管理员 代理 审核员
修改	添加、删除和更新 ACL 等效器。	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，任何人都允许此操作。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 界面关联。此 ACL 仅在实例配置期间可用，且 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，并根据序列号检索证书。	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 Summary

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
修改	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 Summary

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
修改	修改原始实例配置。	Allow	企业管理员

D.2.5. certServer.general.configuration

控制对子系统实例的常规配置的访问，包括谁可以查看并编辑 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 Summary

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、服务器统计信息、加密、令牌名称、证书主题名称、证书别名、服务器、CA 证书以及所有用于管理的证书。	Allow	管理员 代理 审核员
修改	修改 LDAP 数据库、SMTP 和加密的设置。发布导入证书、安装证书、信任和不信任 CA 证书、导入跨修复证书和删除证书。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。根据需要运行自助范围。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名。	Allow	管理员

D.2.6. certServer.log.configuration

控制对证书管理器的日志配置的访问，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 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	查看日志插件信息、日志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
修改	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

限制访问以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
修改	更改日志实例的 **fileName** 参数的值。	拒绝	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.signedAudit

控制有权访问已签名的审计日志的人员。默认设置为：

> *allow (read) group="Auditors"*

表 D.10. 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.2.10. certServer.registry.configuration

控制对管理 registry 的访问，即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.11. certServer.registry.configuration ACL Summary

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

操作	描述	allow/Deny Access	目标用户/组
修改	注册单个配置集实现插件。	Allow	管理员

D.3. 特定于证书的 ACL

本节涵盖了为证书管理器特别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。

为每个 CA 接口（管理控制台和代理和终端服务页面）设置了访问控制规则，以及用于列出和下载证书等常见操作。

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.12. certServer.admin.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	修改 OCSP 配置、OCSP 存储配置和默认 OCSP 存储。	Allow	Enterprise OCSP 管理员
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.13. 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

操作	描述	allow/Deny Access	目标用户/组
unrevoke	更改证书的状态与撤销。	Allow	证书管理器代理
撤销	将证书的状态更改为撤销。	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细信息。	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.14. certServer.ca.certificates ACL Summary

操作	描述	allow/Deny Access	目标用户/组
撤销	吊销证书或批准证书撤销请求。从 TPS 吊销证书。提示用户输入有关撤销请求的额外数据。	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大量证书的详细信息。	Allow	证书管理器代理 注册管理器代理

D.3.4. certServer.ca.configuration

控制证书管理器的常规配置的操作。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.15. certServer.ca.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 CRL 插件信息、常规 CA 配置、CA 连接器配置、CRL 发布点配置、CRL 配置集配置、请求通知配置、撤销通知配置、队列通知配置和 CRL 扩展配置。列出 CRL 扩展配置和 CRL 发布点配置。	Allow	管理员 代理 审核员
修改	添加和删除 CRL 发布点。修改常规 CA 设置、CA 连接器配置、CRL 发布点配置、CRL 配置、请求通知配置、撤销通知配置、队列通知配置和 CRL 扩展配置的请求。	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.16. certServer.ca.connector ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	提交来自远程可信管理器的请求。	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 自动连接来执行密钥归档和恢复操作。这些信任关系通过特殊的连接器插件进行配置。

> *allow (read) group="Enterprise KRA Administrators";allow (modify) group="Enterprise KRA Administrators" || group="Subsystem Group"*

表 D.17. 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

操作	描述	allow/Deny Access	目标用户/组
修改	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

控制通过代理服务接口读取或更新 CRL 的访问。默认设置为：

```
allow (read,update) group="Certificate Manager Agents"
```

表 D.18. certServer.ca.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	显示 CRL 并获取有关 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.19. 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

控制对内部数据库的访问，以便为证书管理器实例添加用户和组。

```
allow (modify,read) group="Administrators"
```

表 D.20. certServer.ca.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	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

通过代理服务接口控制访问和读取 OCSP 信息的能力，如用量统计。

```
allow (read) group="Certificate Manager Agents"
```

表 D.21. 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

在代理服务页面中控制对证书配置文件配置的访问。

```
allow (read,approve) group="Certificate Manager Agents"
```

表 D.22. certServer.ca.profile ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看证书配置文件的详情。	Allow	证书管理器代理
批准	批准并启用证书配置文件。	Allow	证书管理器代理

D.3.12. certServer.ca.profiles

控制在代理服务接口中列出证书配置文件的访问。

> *allow (list) group="Certificate Manager Agents"*

表 D.23. certServer.ca.profiles ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	列出证书配置文件。	Allow	证书管理器代理

D.3.13. certServer.ca.registerUser

定义哪些组或用户可以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.24. certServer.ca.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	注册新代理。	Allow	企业管理员
读取	阅读现有的代理信息。	Allow	企业管理员

D.3.14. certServer.ca.request.enrollment

控制注册请求是如何处理和分配的。默认设置为：

> *allow (submit) user="anybody";allow (read,execute,assign,unassign) group="Certificate Manager Agents"*

表 D.25. 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
执行	修改请求的批准状态。	Allow	证书管理器代理
提交	Sumbit 一个请求。	Allow	anybody
分配	为证书管理器代理分配请求。	Allow	证书管理器代理

操作	描述	allow/Deny Access	目标用户/组
取消分配	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处理。默认设置为：

```
allow (approve,read) group="Certificate Manager Agents"
```

表 D.26. certServer.ca.request.profile ACL Summary

操作	描述	allow/Deny Access	目标用户/组
批准	修改基于证书配置文件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

控制可以在代理服务界面中列出证书请求的人员。

```
allow (list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

表 D.27. certServer.ca.requests ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	检索一系列请求的详细信息，并使用复杂的过滤器搜索证书。	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.28. certServer.ca.systemstatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看统计信息。	Allow	证书管理器代理

D.3.18. certServer.ee.certchain

控制在终端实体页面中可以访问 CA 证书链的人员。

```
allow (download,read) user="anybody"
```

表 D.29. 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

通过终端实体页面控制谁可以访问证书，适用于导入或撤销证书等大多数操作。

```
allow (renew,revoke,read,import) user="anybody"
```

表 D.30. certServer.ee.certificate ACL Summary

操作	描述	allow/Deny Access	目标用户/组
续订	提交请求以续订现有证书。	Allow	任何人
撤销	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

控制可以在终端实体页面中列出撤销的证书或提交撤销请求。

allow (revoke,list) user="anybody"

表 D.31. certServer.ee.certificates ACL Summary

操作	描述	allow/Deny Access	目标用户/组
撤销	提交要撤销的证书列表。	Allow	要撤销的证书主题必须与提供的证书匹配才能向 CA 进行身份验证。
list	搜索与指定条件匹配的证书。	Allow	任何人

D.3.21. certServer.ee.crl

通过终端实体页面控制对 CRL 的访问。

allow (read,add) user="anybody"

表 D.32. 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	在 OCSP 服务器中添加 CRL。	Allow	任何人

D.3.22. certServer.ee.profile

控制对终端实体页面中证书配置文件的一些访问，包括谁可以查看配置文件的详情或通过配置文件提交请求。

allow (submit,read) user="anybody"

表 D.33. certServer.ee.profile ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	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

控制在终端实体页面中列出活跃证书配置文件的人员。

```
allow (list) user="anybody"
```

表 D.34. certServer.ee.profiles ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	列出证书配置文件。	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.35. certServer.ee.request.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	提交 OCSP 请求。	Allow	所有 IP 地址

D.3.25. certServer.ee.request.revocation

控制用户可以在终端实体页面中提交证书撤销请求。

```
allow (submit) user="anybody"
```

表 D.36. certServer.ee.request.revocation ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	提交用于吊销证书的请求。	Allow	任何人

D.3.26. certServer.ee.requestStatus

控制在终端实体页面中查看证书请求的状态。

> *allow (read) user="anybody"*

表 D.37. certServer.ee.requestStatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	检索针对该请求发布的任何证书的请求和序列号的状态。	Allow	任何人

D.3.27. certServer.job.configuration

控制可以为证书管理器配置作业的人员。

> *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.38. 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
修改	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

控制对证书配置文件配置的访问。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.39. certServer.profile.configuration ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看证书配置文件默认设置和约束、输入、输出配置、输出配置、默认配置、策略限制配置和证书配置文件实例配置。列出证书配置文件插件和证书配置文件实例。	Allow	管理员 代理 审核员
修改	添加、修改和删除证书配置文件默认值和约束、输入、输出和证书配置文件实例。添加和修改默认策略约束配置。	Allow	管理员

D.3.29. certServer.publisher.configuration

控制谁可以查看证书管理器的发布配置。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.40. certServer.publisher.configuration ACL Summary

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，以及映射器插件和实例。	Allow	管理员 代理 审核员
修改	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、映射器实例、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。安全域配置在配置期间由子系统实例直接访问和修改，因此必须始终允许适当的访问子系统，或者配置可能会失败。

```
allow (read) user="anybody";allow (modify) group="Subsystem Group"
```

表 D.41. 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	anybody
修改	通过更改实例信息并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. 密钥恢复特定于授权的 ACL

本节介绍针对 KRA 的默认访问控制配置。KRA ACL 配置还包括 第 D.2 节 “常见 ACL” 中列出的所有常用 ACL。

为每个 KRA 接口（管理控制台和代理和终端服务页面）设置了访问控制规则，以及用于列出和下载密钥等常见操作。

D.4.1. certServer.job.configuration

控制可以为 KRA 配置作业的人员。

```
allow (read) group="Administrators" || group="Key Recovery Authority Agents" ||
group="Auditors";allow (modify) group="Administrators"
```

表 D.42. 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
修改	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.43. 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

控制谁可以配置和管理 KRA 的设置。

```
allow (read) group="Administrators" || group="Auditors" || group="Key Recovery Authority Agents" ||
allow (modify) group="Administrators"
```

表 D.44. certServer.kra.configuration ACL Summary

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
修改	更改所需恢复代理批准的数量。	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.45. certServer.kra.connector ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	提交一个新的密钥归档请求（仅限非TMS）。	Allow	可信管理器

D.4.5. certServer.kra.GenerateKeyPair

控制可以将密钥恢复请求提交到 KRA。默认配置是：

```
allow (execute) group="Key Recovery Authority Agents"
```

表 D.46. 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 代理

D.4.6. certServer.kra.getTransportCert

控制可以将密钥恢复请求提交到 KRA。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.47. 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.48. certServer.kra.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	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、恢复或下载密钥来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.49. certServer.kra.key ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	显示有关密钥归档记录的公钥信息。	Allow	KRA 代理
recover	从数据库检索密钥信息，以执行恢复操作。	Allow	KRA 代理
下载	通过代理服务页面下载密钥信息。	Allow	KRA 代理

D.4.9. certServer.kra.keys

控制谁可以通过代理服务页面列出存档的密钥。

> *allow (list) group="Key Recovery Authority Agents"*

表 D.50. certServer.kra.keys ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	搜索并列出一系列存档的键。	Allow	KRA 代理

D.4.10. certServer.kra.registerUser

定义哪些组或用户可以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.51. certServer.kra.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	注册新用户。	Allow	企业管理员
读取	阅读现有用户信息。	Allow	企业管理员

D.4.11. certServer.kra.request

控制谁可以在代理服务接口中查看密钥归档和恢复请求。

```
allow (read) group="Key Recovery Authority Agents"
```

表 D.52. 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 代理

D.4.12. certServer.kra.request.status

控制在终端实体页面中查看密钥恢复请求的状态。

```
allow (read) group="Key Recovery Authority Agents"
```

表 D.53. 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 代理

D.4.13. certServer.kra.requests

控制可以在代理服务界面中列出密钥归档和恢复请求。

```
allow (list) group="Key Recovery Authority Agents"
```

表 D.54. certServer.kra.requests ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	检索一系列密钥存档和恢复请求的详细信息。	Allow	KRA 代理

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.55. certServer.kra.systemstatus ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看统计信息。	Allow	KRA 代理

D.4.15. certServer.kra.TokenKeyRecovery

控制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.56. certServer.kra.TokenKeyRecovery ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	为令牌恢复提交或启动密钥恢复请求。	Allow	KRA 代理

D.5. 特定于在线证书状态管理器的 ACL

本节介绍为在线证书状态管理器设置的默认访问控制配置属性。OCSP 响应器的 ACL 配置还包括 *第 D.2 节 “常见 ACL” 中列出的所有通用 ACL。*

为每个 OCSP 接口（管理控制台和代理和终端服务页面）设置了访问控制规则，以及用于列出和下载 CRL 等常见操作。

D.5.1. certServer.ee.crl

通过终端实体页面控制对 CRL 的访问。

```
allow (read) user="anybody"
```

表 D.57. 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.58. certServer.ee.request.ocsp ACL Summary

操作	描述	allow/Deny Access	目标用户/组
提交	提交 OCSP 请求。	Allow	所有 IP 地址

D.5.3. certServer.ocsp.ca

控制谁可以指示 OCSP 响应器。默认设置为：

```
allow (add) group="Online Certificate Status Manager Agents"
```

表 D.59. certServer.ocsp.ca ACL Summary

操作	描述	allow/Deny Access	目标用户/组
添加	指示 OCSP 响应响应新 CA 的 OCSP 请求。	Allow	OCSP Manager 代理

D.5.4. certServer.ocsp.cas

控制在代理服务界面中列出哪些用户可以向在线证书状态管理器发布 CRL 的所有证书管理器。默认设置为：

> allow (list) group="Online Certificate Status Manager Agents"

表 D.60. certServer.ocsp.cas ACL Summary

操作	描述	allow/Deny Access	目标用户/组
list	列出所有向 OCSP 响应者发布 CRL 的证书管理器。	Allow	代理

D.5.5. certServer.ocsp.certificate

控制谁可以验证证书的状态。默认设置为：

> allow (validate) group="Online Certificate Status Manager Agents"

表 D.61. 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 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.62. certServer.ocsp.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 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
修改	修改 OCSP 配置、OCSP 存储配置和默认 OCSP 存储。	Allow	管理员

D.5.7. certServer.ocsp.crl

控制通过代理服务接口读取或更新 CRL 的访问。默认设置为：

> *allow (add) group="Online Certificate Status Manager Agents" || group="Trusted Managers"*

表 D.63. certServer.ocsp.crl ACL Summary

操作	描述	allow/Deny Access	目标用户/组
add	向由 OCSP 响应者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

控制为在线证书状态管理器实例添加用户和组的内部数据库的访问。

> *allow (modify,read) group="Administrators"*

表 D.64. certServer.ocsp.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	创建、编辑或删除实例的用户和组条目。	Allow	管理员

操作	描述	allow/Deny Access	目标用户/组
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 响应器的信息。

```
allow (read) group="Online Certificate Status Manager Agents"
```

表 D.65. certServer.ocsp.info ACL Summary

操作	描述	allow/Deny Access	目标用户/组
读取	查看 OCSP 响应者信息。	Allow	OCSP 代理

D.6. 令牌密钥服务特定 ACL

本节涵盖为令牌密钥服务(TKS)特别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” 中列出的所有常用 ACL。

为 TKS 的管理控制台设置了访问控制规则，并可通过其他子系统访问 TKS。

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.66. 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.67. certServer.tks.group ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	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 TKS 用来发送密钥的传输证书。

```
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.68. certServer.tks.importTransportCert ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	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

控制谁可以查看由 TKS 派生和存储的密钥集合的信息。

```
allow (execute) group="Token Key Service Manager Agents"
```

表 D.69. certServer.tks.keysetdata ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	创建分离的密钥集数据。	Allow	TKS 代理

D.6.5. certServer.tks.registerUser

定义哪些组或用户可以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.70. certServer.tks.registerUser ACL Summary

操作	描述	allow/Deny Access	目标用户/组
修改	注册新代理。	Allow	企业管理员
读取	阅读现有的代理信息。	Allow	企业管理员

D.6.6. certServer.tks.sessionkey

控制谁可以创建 TKS 实例连接到 TPS 的会话密钥。

```
allow (execute) group="Token Key Service Manager Agents"
```

表 D.71. certServer.tks.sessionkey ACL Summary

操作	描述	allow/Deny Access	目标用户/组
执行	创建 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.72. 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

本附录提供单独的审计事件及其参数描述和格式。日志中的每个审计事件都由以下信息组成：

- 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 为 SIGNED_AUDIT)：

  > [14]

- 当前的日志级别(6)与安全相关的事件。请参阅 Red Hat Certificate System 规划、安装和部署指南中的 日志级别(Message Categories) 部分。例如：

  > [6]

- 有关日志事件（这是特定日志事件）的信息；有关特定日志事件中的每个字段的信息，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

以下列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

一个

Administrator

安装和配置一个或多个证书系统管理器并为他们设置特权用户或代理的人员。另请参阅 agent。

agent

属于组的用户，授权为证书系统管理器管理 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。通信单元（与字节相同），用于智能卡和智能卡读卡器之间的通信。

代理批准的注册

在签发证书前需要代理批准请求的注册。

代理服务

1.可以通过由证书系统 agent 通过为代理分配所需权限的证书系统子系统提供的 HTML 页面管理的服务。

2.用于管理这些服务的 HTML 页面。

审核员

此特权用户可以查看签名的审计日志。

审计日志

记录各种系统事件的日志。此日志可以被签名，提供未被篡改的证明，且只能由审核员用户读取。

属性值断言(AVA)

form 属性 = value 的断言，其中 attribute 是标签，如 o （机构）或 uid （用户 ID），value 是一个值，如 "Red Hat, Inc." 或登录名称。AVAs 被用来组成用来标识证书主题的 可区分名称(DN)，称为证书的 主题名称。

授权

访问服务器控制的资源的权限。通常，在与资源关联的 ACL 由服务器评估后，通常会进行授权。请参阅 访问控制列表(ACL)。

自动注册

配置证书系统子系统的方法，允许自动身份验证最终用户注册，而无需人为干预。通过这种身份验证形式，可以成功完成身份验证模块处理的证书请求被自动批准，以进行配置文件处理和证书颁发。

访问控制

控制允许哪些特定用户执行的操作。例如，对服务器的访问控制通常基于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)

一组访问控制条目，用于定义服务器收到对特定资源访问请求时要评估的访问规则的层次结构。请参阅 访问控制指令(ACI)。

访问控制指令(ACI)

指定请求访问权限的主体如何识别或拒绝特定主题的访问权限的访问规则。请参阅 访问控制列表(ACL)。

身份验证

满意识别；保证对某些计算机事务的方并不是一个不可变的发布者。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（作为 Java™ 类）来验证终端实体、代理、管理员或其他需要与证书系统子系统交互的其他实体。对于典型的最终用户注册，用户在用户提供注册表单请求的信息后，注册 servlet 会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)类似它的数据加密标准(DES)，是一个 FIPS 批准的对称密钥加密标准。AES 被美国政府在 2002 年采用。它定义了三个块密码、AES-128、AES-192 和 AES-256。国家标准与技术研究机构(NIST)在美国定义 AES 标准。FIPS PUB 197。有关更多信息，请参阅 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf。

B

绑定 DN

用户 ID，格式为可分辨名称(DN)，用于与 Red Hat Directory Server 进行身份验证的密码。

C

CA 层次结构

root CA 将证书委派给从属 CA 的 CA 层次结构。子级 CA 也可以通过将状态委派给其他 CA 来扩展层次结构。另请参阅 证书颁发机构(CA)、子 CA、根 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 使用其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)、子 CA、根 CA。

certificate

根据 X.509 标准格式化的数字数据，指定个人、公司或其他实体（证书的 主题名称 ）的名称，证书也包含在证书中，该实体也属于该实体。公钥证书由 证书颁发机构(CA) 发布并数字签名。可以通过 数字签名 技术检查 CA 的 公钥加密 来验证证书的有效性。要在 公钥基础架构(PKI) 中信任，证书必须由在 PKI 中注册的其他实体信任的 CA 发布并签名。

CMC

请参阅 Cryptographic Message Syntax (CMC)上的证书管理消息。

CMC 注册

允许签名注册或签名撤销请求使用代理的签名证书发送到证书管理器的功能。然后，证书管理器会自动处理这些请求。

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)。

Cryptographic Message Syntax (CMC)上的证书管理消息

用于将证书请求传递给证书管理器的消息格式。来自互联网工程任务强制(IETF) PKIX 工作组的提议标准。如需更多信息，请参阅 https://tools.ietf.org/html/draft-ietf-pkix-cmc-02。

CSP

请参阅 加密服务提供商(CSP)。

信任链

请参阅 证书链。

加密服务提供商(CSP)

一个加密模块，它代表使用 PKCS 定义的标准接口，如密钥生成、密钥存储和加密等加密服务。

加密模块

请参阅 PKCS the module。

加密消息语法(CS)

用于数字签名、摘要、验证或加密任意消息的语法，如 CMMF。

加密算法

用于执行加密操作的一组规则或指示，如 encryption 和 解密。

基于证书的验证

根据证书和公钥加密进行身份验证。另请参阅 基于密码的身份验证。

客户端 SSL 证书

用于识别使用 SSL 协议到服务器的证书。请参阅 安全套接字层(SSL)。

客户端身份验证

将客户端标识到服务器的过程，如使用名称和密码，或者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。

密码

请参阅 *加密算法*。

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其他字段。证书扩展提供了一种向证书添加信息（如备用主题名称和用量限制）的方法。PKIX 工作组定义了多个标准扩展。

证书指纹

与证书关联的 单向哈希。这个值不是证书本身的一部分，而是通过将哈希功能应用到证书的内容来生成。如果证书的内容发生了变化，即使单个字符也是如此，则同一函数也会生成不同的数字。因此，可以使用证书指纹来验证证书是否未被篡改。

证书撤销列表(CRL)

由 X.509 标准定义，按序列号吊销的证书列表，由 证书颁发机构(CA) 生成并签名。

证书管理器

用作证书颁发机构的独立证书系统子系统。证书管理器实例问题、续订和撤销证书，它可以与 CRL 一起发布到 LDAP 目录。它接受来自结束实体的请求。请参阅 证书颁发机构(CA)。

证书管理器代理

属于管理证书管理器代理服务的组授权的用户。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能力。

证书管理消息格式(CMMF)

用来将证书请求和撤销从终端实体发送到证书管理器的消息格式，并将各种信息发送到最终实体。来自互联网工程任务强制(IETF) PKIX 工作组的提议标准。CMMF 被另一个提议的标准 Cryptographic Message Syntax (CMC)上的证书管理消息 使用。如需更多信息，请参阅 https://tools.ietf.org/html/draft-ietf-pkix-cmmf-02。

证书系统

请参阅 *Red Hat Certificate System*、*加密消息语法(CS)*。

证书系统子系统

五个证书系统管理器之一： 证书管理器、在线证书状态管理器、令牌密钥服务或令牌处理系

统。密钥恢复授权

证书系统控制台

可以为任何单一证书系统实例打开的控制台。证书系统控制台允许证书系统控制相应证书系统实例的配置设置。

证书请求消息格式(CRMF)

用于管理 X.509 证书的消息的格式。这个格式是 CMMF 的子集。另请参阅 证书管理消息格式(CMMF)。如需更多信息，请参阅 https://tools.ietf.org/html/rfc2511。

证书配置文件

定义特定类型的注册的一组配置设置。证书配置文件为特定类型的注册设置策略，以及证书配置文件中的身份验证方法。

证书链

由连续证书颁发机构签名的一系列证书。CA 证书标识了一个 证书颁发机构(CA)，用于签署该机构发布的证书。CA 证书可反过由父 CA 的 CA 证书签名，以此类推 根 CA。证书系统允许任何最终实体检索证书链中的所有证书。

证书颁发机构(CA)

在验证证书要识别的人员或实体的身份后，发出 certificate 的可信实体。CA 还续订并撤销证书并生成 CRL。在证书的 issuer 字段中命名的实体始终是一个 CA。证书颁发机构可以使用证书颁发的服务器软件（如 Red Hat Certificate System）的独立第三方或机构。

跨对证书

一个 CA 发布的证书到另一个 CA，然后由两个 CA 存储，以此组成一个信任的圆圈。这两个 CA 相互发布证书，然后将跨修复证书存储为证书对。

跨技术

按不同认证层次结构中的两个 CA 或链中的证书交换。跨技术扩展了信任的链，使其包含这两个层次结构。另请参阅 证书颁发机构(CA)。

链 CA

请参阅 链接的 CA。

D

delta CRL

包含自上次完整 CRL 后撤销的这些证书的 CRL 列表。

双密钥对

两个公钥-私钥对，四个密钥都对应两个单独的证书。一个对的私钥用于签名操作，另一个对的公钥和私钥用于加密和解密操作。每个对都对应一个独立的 certificate。另请参阅 加密密钥、公钥加密、签名密钥。

发布点

用于 CRL 来定义一组证书。每个发行版点都由一组发布的证书定义。可以为特定的发布点创建 CRL。

可区分名称(DN)

一系列识别证书主题的 AVAs。请参阅 属性值断言(AVA)。

密钥恢复授权

一个可选的独立证书系统子系统，用于管理用于结束实体的 RSA 加密密钥的长期归档和恢复。证书管理器可以配置为在发布新证书前，使用密钥恢复授权来归档最终实体的加密密钥。只有当最终实体加密数据（如敏感电子邮件）时，Key Recovery Authority 才很有用，组织可能需要每天恢复。它只能用于支持双密钥对的结束实体：两个单独的密钥对，一个用于加密，另一个用于数字签名。

密钥恢复授权代理

属于授权管理密钥恢复授权机构代理服务的用户，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。

密钥恢复授权存储密钥

密钥恢复授权机构使用的特殊密钥加密最终实体的加密密钥，在使用密钥恢复授权机构的私钥解密后加密它。存储密钥永远不会离开密钥恢复授权。

密钥恢复授权恢复代理

拥有部分存储密钥的 n 人之一 密钥恢复授权。

密钥恢复授权机构传输证书

认证最终实体使用的公钥，以加密实体的加密密钥，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，在使用存储密钥加密之前解密最终实体的密钥。

数字 ID

请参阅 certificate。

数字签名

要创建数字签名，签名软件首先从要签名的数据（如新签发的证书）创建一个 单向哈希。然后，单向哈希使用签名人的私钥进行加密。生成的数字签名对于签名的每个数据都是唯一的。即使单个逗号添加到消息中也会更改该消息的数字签名。使用签名人的公钥成功解密数字签名，并与同一数据的另一个哈希进行比较，提供 篡改检测。为包含公钥的证书验证 证书链，提供签名人验证。另请参阅 nonRepudiation、encryption。

解密

未处理已加密的数据。请参阅 encryption。

E

eavesdropping

Surreptitious 截获通过网络发送的信息，由信息不预期的实体发送。

elliptic Curve Cryptography (ECC)

使用 elliptic curves 为数学问题创建附加日志变体的加密算法，这是加密密钥的基础。ECC 密码比 RSA 密码更高效，并且由于其内部复杂性比 RSA 密码更强。

encryption

以忽略其含义的方式处理信息。请参阅 解密。

extensions 字段

请参阅 证书扩展。

加密密钥

仅用于加密的私钥。加密密钥及其等同的公钥，以及一个 签名密钥 及其等同的公钥组成一个 双密

钥对。

注册

请求和接收 X.509 证书以便在 公钥基础架构(PKI) 中使用的过程。也称为 注册。

结束实体

在 公钥基础架构(PKI) 中，个人、路由器、服务器或其他使用 certificate 识别其自身的实体。

F

fingerprint

请参阅 证书指纹。

FIPS PUBS 140

联邦信息处理标准(FIPS PUBS) 140 是加密模块、硬件或软件实现的美国政府标准，用于加密和解密数据或执行其他加密操作，如创建或验证数字签名。销售到美国政府的许多产品都必须符合一个或多个 FIPS 标准。请参阅 http://www.nist.gov。

firewall

在两个或多个网络间强制实施边界的系统或组合。

联邦信息处理标准证书颁发机构(FBCA)

两个 CA 形成一个信任的配置，方法是向彼此发布跨对证书，并将两个跨密钥对存储为单一证书对。

I

IP 欺骗

客户端 IP 地址的伪造。

中间 CA

其证书位于 root CA 和 证书链 中发布的证书的 CA。

模拟

将 posing 作为通过网络发送的信息的预期接收者。模拟可以采用两种形式： 欺骗 和 错误代表。

输入

在证书配置文件功能上下文中，它会为特定证书配置文件定义注册表单。设置每个输入，然后从为此注册配置的所有输入动态创建注册表单。

J

JAR 文件

为压缩的文件集合进行数字信封，根据 Java™ 归档(JAR)格式 进行组织。

Java™ Development Kit (JDK)

由 Sun Microsystems 提供的软件开发工具包，用于使用 Java™ 编程语言开发应用程序和小程序。

Java™ 加密架构(JCA)

由 Sun Microsystems 为加密服务开发的 API 规格和引用。See http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.Introduction.

Java™ 原生接口(JNI)

在给定平台上提供 Java™ 虚拟机(JVM)不同实现的标准编程接口，允许使用 C 或 C++ 等语言编写的现有代码，以便单一平台绑定到 Java™。请参阅 http://java.sun.com/products/jdk/1.2/docs/guide/jni/index.html。

Java™ 安全服务(JSS)

用于控制由网络安全服务(NSS)执行的安全操作的 Java™ 接口。

Java™ 归档(JAR)格式

用于将数字签名、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 密钥交换算法(KEA)。

key

加密算法 用来加密或解密数据的大量数字。例如，个人的 公钥 允许其他人加密针对该人员的信息。然后，必须使用对应的 私钥 解密信息。

密钥交换

然后有一个客户端和服务器来确定它们在 SSL 会话中使用的对称密钥。

密钥交换算法(KEA)

用于美国政府密钥交换的算法。

L

轻量级目录访问协议(LDAP)

一个目录服务协议，旨在通过 TCP/IP 和多个平台运行。LDAP 是目录访问协议(DAP)的简化版本，用于访问 X.500 目录。LDAP 处于 IETF 更改控制，并已扩展以满足互联网要求。

链接的 CA

一个内部部署的 证书颁发机构(CA)，它的证书由公共的第三方 CA 签名。内部 CA 充当它发布的证书的 root CA，第三方 CA 充当链接到同一第三方 root CA 的其他 CA 发布的证书的根 CA。也称为"链 CA"以及由不同公共 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。

手动身份验证

配置需要人批准每个证书请求的证书系统子系统的方法。借助这种身份验证形式，servlet 会在身份验证模块处理后将证书请求转发到请求队列。具有适当权限的代理必须在配置文件处理和证书颁发前单独批准每个请求。

消息摘要

请参阅 单向哈希。

错误代表

以个人或机构形式的实体演示。例如，当网站实际上是需要信用卡付款的网站，但永远不会发送任何好的网站时，网站可能会认为是模糊存储。Misrepresentation 是 模拟 的一种形式。另请参阅 欺骗。

N

non-TMS

非令牌管理系统。指的是 不直接 处理智能卡的子系统(CA 以及可选的 KRA 和 OCSP)的配置。

参见令牌管理系统.

nonRepudiation

消息发送者无法拒绝发送邮件。数字签名 提供了一种非缓解形式。

网络安全服务(NSS)

一组库，旨在支持启用了安全的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、篡改检测和加密令牌接口，以及用于加密令牌接口的 PKCSROX 协议。NSS 也作为软件开发工具包单独提供。

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

在证书配置文件功能上下文中，它会定义从成功注册特定证书配置文件中的生成的表单。设置每个输出，然后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，并帮助哈希算法。数字（也称为消息摘要）对于哈希数据是唯一的。数据的任何更改（甚至删除或更改单个字符）都会生成不同的值。

2.哈希数据的内容不能从散列中分离。

对象签名

一个文件签名方法，使软件开发人员能够签署 Java 代码、JavaScript 脚本或任何类型的文件，并允许用户识别签名代码对本地系统资源的访问。

对象签名证书

关联的私钥证书用于为对象签名；与 对象签名 相关的证书。

P

PKCS #10

管理证书请求的公钥加密标准。

PKCS #12

管理关键可移植性的公钥加密标准。

PKCS #7

管理签名和加密的公钥加密标准。

PKCS THE

管理加密令牌（如智能卡）的公钥加密标准。

PKCS the module

通过 PKCS the 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 the 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 the 模块始终具有一个或多个插槽，它们可能以某种物理读取器的形式作为物理硬件插槽实施，如智能卡，或软件中的概念插槽。PKCS the module 的每个插槽都可以包含一个令牌，即实际提供加密服务的硬件和软件设备，以及可选的存储证书和密钥。红帽使用证书系统提供了内置的 PKCS the 模块。

公钥

公钥加密中使用的一对密钥。公钥是免费发布，并作为 certificate 的一部分发布。它通常用于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。

公钥加密

一组良好的技术和标准，允许实体以电子方式验证其身份或加密电子数据。涉及两个密钥，一个公钥和一个私钥。公钥 作为证书的一部分发布，该证书将该密钥与特定身份相关联。对应的私钥会保留 secret。使用公钥加密的数据只能使用私钥解密。

公钥基础架构(PKI)

有助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

通过名称和密码来自信识别。另请参阅 身份验证、基于证书的验证。

概念验证(POA)

使用私钥恢复授权传输密钥签名的数据，其中包含有关存档最终用户密钥的信息，包括密钥序列号、密钥恢复机构的名称、对应证书的 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归档操作后向证书颁发机构返回的响应。另请参阅 密钥恢复授权机构传输证书。

私钥

公钥加密中使用的一对密钥。私钥已保存 secret，用于解密使用对应 公钥 加密的数据。

R

RC2、RC4

由 Rivest 为 RSA 数据安全开发的加密算法。另请参阅 加密算法。

Red Hat Certificate System

用于创建、部署和管理证书的高度可配置的软件组件和工具。证书系统由五个主要子系统组成，可在不同物理位置的不同证书系统实例中安装： 证书管理器、在线证书状态管理器、令牌密钥服务和令牌处理系统。密钥恢复授权

RSA 密钥交换

基于 RSA 算法的 SSL 的 key-exchange 算法。

RSA 算法

Rivest-Shamir-Adleman 是用于加密和验证的公钥算法的缩写。它由 Ronald Rivest、Adi Shamir 和 Leonard Adleman 开发，并在 1978 中引入。

根 CA

证书颁发机构(CA)，在证书链的顶部带有一个自签名证书。另请参阅 CA 证书、子 CA。

注册

请参阅 注册。

S

sandbox

Java™ 术语用于定义 Java™ 代码必须在其中操作的限制。

servlet

代表证书系统子系统处理特定类型的与最终实体的 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

安全哈希算法，美国政府使用的哈希函数。

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 字段包含一个唯一描述认证实体的 主题名称。

主题名称

唯一描述了 可区分名称(DN) 的 subject 的 certificate。

单点登录

1.在证书系统中，通过为内部数据库和令牌存储密码，简化了签署红帽证书系统的密码。每次用户登录时，都需要输入此单一密码。

2.用户能够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可以采用多种形式，包括自动跟踪与不同服务器一起使用的密码的机制。证书支持 公钥基础架构 (PKI) 中的单点登录。用户可以一次登录到本地客户端的私钥数据库，只要客户端软件正在运行，则依赖 基于证书的验证 访问用户可访问的机构中的每个服务器。

子 CA

证书由另一个从属 CA 或 root CA 签名的证书颁发机构。请参阅 CA 证书、根 CA。

安全域

PKI 子系统的集中存储库或清单。它的主要用途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)

允许客户端和服务器间双向身份验证的协议，以及经过验证和加密的连接建立。SSL 在 TCP/IP 上运行，并在 HTTP、LDAP、IIl、NNTP 和其他高级别网络协议之上运行。

安全频道

TPS 和智能卡之间的安全关联，允许根据 TKS 和智能卡 APDU 生成的共享主密钥进行加密。

对称加密

使用相同的加密密钥来加密和解密给定消息的加密方法。

插槽

PKCS the module 的一部分，在硬件或软件中实施，其中包含 token。

智能卡

包含微处理器并存储加密信息（如密钥和证书）的小设备，并执行加密操作。智能卡实现一些或者所有 PKCS THE 接口。

服务器 SSL 证书

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

将服务器标识到客户端的过程。另请参阅 客户端身份验证。

欺骗

认为是其他人。例如，个人可以预先使用电子邮件地址 jdoe@example.com，或者计算机可以将自身识别为称为 www.redhat.com 的网站。欺骗是 模拟 的一种形式。另请参阅 错误代表。

签名密钥

仅用于签名的私钥。签名密钥及其等同的公钥，加上 加密密钥 及其等同的公钥构成了 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。证书系统支持 MD5 和 SHA 签名算法。另请参阅 加密算法、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一个签名证书，其公钥对应于它用来签署其问题的证书。

自我测试

在实例启动和按需测试证书系统实例时测试的功能。

T

token

与 插槽 中的 PKCS the module 关联的硬件或者软件设备。它提供加密服务，并选择性地存储证书和密钥。

trust

高度依赖于个人或其他实体。在 公钥基础架构(PKI) 中，信任是指证书用户和签发证书的 证书颁发机构(CA) 之间的关系。如果 CA 是可信的，则该 CA 发布的有效证书可以被信任。

令牌处理系统(TPS)

直接交互企业安全客户端和智能卡的子系统，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，它根据智能卡 APDU 和其他共享信息（如令牌 CUID）为每个智能卡生成特定的独立密钥。

令牌管理系统(TMS)

相互相关的子系统 - CA、TKS、TPS 和可选的 KRA - 用于管理智能卡（令牌）上的证书。

树层次结构

LDAP 目录的层次结构。

篡改检测

确保以电子格式接收的数据与同一数据的原始版本完全对应。

V

虚拟专用网络(VPN)

连接企业地理位置的途径。VPN 允许划分通过加密频道进行通信，允许身份验证的、通常仅限于专用网络的机密事务。

索引

符号

下载证书，在证书系统数据库中安装证书

为什么撤销证书，吊销证书的原因

主机名

用于通知的邮件服务器，为证书系统通知配置邮件服务器

代理

修改

组成员身份，更改组中的成员

创建，创建用户

删除，删除证书系统用户

另请参阅代理服务接口，代理

在个人中注册用户，证书撤销页面

角色定义，代理

代理证书

Request （请求），通过最终用户页面请求和接收证书

令牌

更改密码，更改令牌的密码

查看安装了哪些令牌，查看令牌

管理，管理子系统使用的令牌

令牌密钥服务

代理

创建，创建用户

管理员

创建，创建用户

令牌的子系统

企业安全客户端，证书系统子系统的审阅

令牌管理系统

企业安全客户端，企业安全客户端

任务模块

注册一个新，注册作业模块

企业安全客户端，企业安全客户端

传输证书，传输密钥描述和证书
删除，从数据库中删除证书
更改信任设置，更改 CA 证书的信任设置
查看详情，通过控制台查看数据库内容

位置
活跃日志文件，配置子系统日志

使用的映射程序
CA 证书，LdapCaSimpleMap
DN 组件，LdapDNCompsMap

修改
特权用户的组成员资格，更改组中的成员

停止
子系统实例，启动、停止和重启 PKI 实例
管理员的 sudo 权限，为证书系统服务设置 sudo 权限

内部数据库
schema，配置 LDAP 数据库
名称格式，限制对内部数据库的访问
如何与其他目录服务器实例区分开，限制对内部数据库的访问
它的作用，配置 LDAP 数据库
安装后，配置 LDAP 数据库
已定义，配置 LDAP 数据库
默认主机名，更改内部数据库配置
更改主机名的准备，更改内部数据库配置

删除
发布者模块，注册自定义映射程序和过期插件模块
日志模块，管理日志模块
映射程序模块，注册自定义映射程序和过期插件模块
特权用户，删除证书系统用户
身份验证模块，注册自定义身份验证插件

加密的文件系统(EFS)，扩展密钥用法扩展默认值

发布

CRLs，关于撤销证书
到 LDAP 目录，发布 CRL，LDAP 发布
到文件，发布到文件

queue，启用发布队列
(参见 发布队列)

查看内容，查看证书和 CRL 发布到文件
证书
到文件，发布到文件

发布目录
已定义，LDAP 发布

发布者模块
删除，注册自定义映射程序和过期插件模块
注册一个新，注册自定义映射程序和过期插件模块

发布队列，启用发布队列
启用，启用发布队列

可信管理器
修改
组成员身份，更改组中的成员

删除，删除证书系统用户

可发布到的发布者
files，FileBasedPublisher
OCSP 响应器，OCSPPublisher
目录中的 CA 条目，LdapCaCertPublisher，LdapCrlPublisher，LdapCertificatePairPublisher
目录中的用户条目，LdapUserCertPublisher

吊销证书
原因，吊销证书的原因
谁可以撤销证书，吊销证书的原因

吊销证书的原因，吊销证书的原因

名称扩展模块
签发者备用名称，签发者备用名称扩展默认

启动
子系统实例，启动、停止和重启 PKI 实例
没有 java 安全管理器，启动没有 Java 安全管理器的子系统实例
管理员的 sudo 权限，为证书系统服务设置 sudo 权限

命令行工具
为证书系统证书添加扩展，请求签名证书，请求其他证书

命名规则
对于内部数据库实例，限制对内部数据库的访问

在线证书状态管理器
代理
创建，创建用户

密钥对和证书
SSL 服务器证书，SSL 服务器密钥检测和证书
子系统证书，子系统证书
签名证书，OCSP 签名密钥和证书

管理员
创建，创建用户

基于文件的发布者，FileBasedPublisher
备份证书系统，备份和恢复证书系统
如何撤销证书，吊销证书的原因
子系统证书，子系统证书，子系统证书，子系统证书
nickname，子系统证书，子系统证书，子系统证书
存储密钥对，存储密钥分区
安装证书，在证书系统数据库中安装证书
审核员
创建，创建用户

密钥恢复授权
代理
创建，创建用户

密钥对和证书
传输证书，传输密钥描述和证书

列表，密钥恢复授权证书

子系统证书，子系统证书

存储密钥对，存储密钥分区

管理员

创建，创建用户

归档

轮转日志文件，日志文件轮转

恢复证书系统，备份和恢复实例目录

扩展的密钥使用扩展

加密文件系统的 OID，扩展密钥用法扩展默认值

插件模块

对于 CRL 扩展

CRLReason，Newest CRL 扩展默认值

对于调度作业

unpublishExpiredCerts，unpublishExpiredCerts (UnpublishExpiredJob)

用于发布

FileBasedPublisher，FileBasedPublisher

LdapCaCertPublisher，LdapCaCertPublisher，LdapCertificatePairPublisher

LdapCaSimpleMap，LdapCaSimpleMap

LdapCrlPublisher，LdapCrlPublisher

LdapDNCompsMap，LdapDNCompsMap

LdapUserCertPublisher，LdapUserCertPublisher

OCSPPublisher，OCSPPublisher

签发者备用名称，签发者备用名称扩展默认

日志刷新间隔，缓冲和不缓冲的日志记录

日志模块

删除，管理日志模块

注册一个新，管理日志模块

日志轮转的时间，日志文件轮转

映射程序模块

删除，注册自定义映射程序和过期插件模块

注册一个新，注册自定义映射程序和过期插件模块

更改

组成员，更改组中的成员

证书中的信任设置，更改 CA 证书的信任设置

为什么将改变，更改 CA 证书的信任设置

最终用户证书发布者，LdapUserCertPublisher

未缓冲的日志，缓冲和不缓冲的日志记录

注册

任务模块，注册作业模块

发布者模块，注册自定义映射程序和过期插件模块

启动的代理，证书撤销页面

日志模块，管理日志模块

映射程序模块，注册自定义映射程序和过期插件模块

自定义 OID，标准 X.509 v3 证书扩展参考

身份验证模块，注册自定义身份验证插件

活跃日志

消息类别，正在记录的服务

默认文件位置，配置子系统日志

添加

extensions

CRLs，设置 CRL 扩展

特权用户

修改权限

组成员身份，更改组中的成员

删除，删除证书系统用户

类型

代理，代理

用于通知的邮件服务器，为证书系统通知配置邮件服务器

用户证书

Request （请求），通过最终用户页面请求和接收证书

目录
从中删除过期的证书，unpublishExpiredCerts (UnpublishExpiredJob)

端到端证书
续订，配置配置集以启用续订

签名
轮转日志文件，签名日志文件

签名算法，为证书设置签名算法
ECC 证书，为证书设置签名算法
RSA 证书，为证书设置签名算法

签名证书，OCSP 签名密钥和证书
nickname，OCSP 签名密钥和证书
删除，从数据库中删除证书
更改信任设置，更改 CA 证书的信任设置
查看详情，通过控制台查看数据库内容

管理
证书数据库，管理证书数据库

管理员
sudo 权限，为证书系统服务设置 sudo 权限
修改
组成员身份，更改组中的成员

创建，创建用户
删除，删除证书系统用户
提供的工具
证书系统控制台，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS 子系统

缓冲的日志，缓冲和不缓冲的日志记录
网桥证书，使用跨证书
设置 CRL 扩展，设置 CRL 扩展
证书
发布到 LDAP 目录
所需的模式，配置 LDAP 目录

发布到文件，发布到文件

吊销原因，吊销证书的原因

如何撤销，吊销证书的原因

安装，在证书系统数据库中安装证书

扩展，在 CA 证书中设置限制，证书和 CRL 的默认值、约束和扩展

签名算法，为证书设置签名算法

证书撤销

原因，吊销证书的原因

谁可以撤销证书，吊销证书的原因

身份验证期间，User-Initiated Revocation

证书数据库

在其中维护它，管理证书数据库

如何管理，管理证书数据库

它包含的内容，管理证书数据库

证书管理器

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 签名密钥和证书

OCSP 签名证书，OCSP 签名密钥和证书

SSL 服务器证书，SSL 服务器密钥检测和证书

TLS CA 签名证书，OCSP 签名密钥和证书

子系统证书，子系统证书

序列号范围，在颁发证书上更改 CA 的限制

管理员

创建，创建用户

配置

通知的 SMTP 设置，为证书系统通知配置邮件服务器

证书系统

备份，备份和恢复证书系统

恢复，备份和恢复实例目录

证书系统控制台

Status 标签页，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS 子系统

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 子系统

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

存储的位置，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 安装证书链，通过控制台安装证书

使用安装证书，通过控制台安装证书

证书配置文件

签名算法，为证书设置签名算法

证书链

为什么安装，关于 CA 证书链

在证书数据库中安装，通过控制台安装证书

请求证书

CA 签名证书，通过控制台请求证书

CRL 签名证书，通过控制台请求证书

ECC 证书，创建证书签名请求

KRA 传输证书，通过控制台请求证书

OCSP 签名证书，通过控制台请求证书

SSL 客户端证书，通过控制台请求证书

SSL 服务器证书，通过控制台请求证书

代理证书，通过最终用户页面请求和接收证书

使用 certutil，创建证书签名请求

用户证书，通过最终用户页面请求和接收证书

通过控制台，通过控制台请求证书

通过终端实体页面，通过最终用户页面请求和接收证书

跨修复证书，使用跨证书

身份验证
在证书撤销过程中，User-Initiated Revocation
通过控制台管理，设置基于 PIN 的注册

身份验证模块
代理启动的用户注册，证书撤销页面
删除，注册自定义身份验证插件
注册一个新，注册自定义身份验证插件

轮转日志文件
如何设置时间，日志文件轮转
归档文件，日志文件轮转
签名文件，签名日志文件

过期的证书
从目录中删除，unpublishExpiredCerts (UnpublishExpiredJob)

配置标签页，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS 子系统

重启
子系统实例，启动、停止和重启 PKI 实例
没有 java 安全管理器，启动没有 Java 安全管理器的子系统实例
管理员的 sudo 权限，为证书系统服务设置 sudo 权限

错误日志
已定义，Tomcat 错误和访问日志

A

authorityInfoAccess，authorityInfoAccess

authorityKeyIdentifier，在 CA 证书中设置限制，authorityKeyIdentifier，authorityKeyIdentifier

B

backups，备份和恢复证书系统

base-64 编码文件
查看内容，查看证书和 CRL 发布到文件

basicConstraints，basicConstraints

C

CA
SCEP 设置，为 SCEP 配置安全设置
启用 SCEP 注册，启用 SCEP 注册
配置 ECC 签名算法，为证书设置签名算法

CA 签名证书，CA 签名密钥和证书
nickname，CA 签名密钥和证书
Request （请求），通过控制台请求证书
删除，从数据库中删除证书
更改信任设置，更改 CA 证书的信任设置
查看详情，通过控制台查看数据库内容

CA 证书发布者，LdapCaCertPublisher，LdapCertificatePairPublisher
CA 证书映射器，LdapCaSimpleMap
certificate
查看内容，查看证书和 CRL 发布到文件

certificateIssuer，certificateIssuer
certificatePolicies，certificatePoliciesExt
certutil
请求证书，创建证书签名请求

CRL
查看内容，查看证书和 CRL 发布到文件

CRL publisher，LdapCrlPublisher
CRL 分发点扩展，CRL 颁发点
CRL 扩展模块
CRLReason，Newest CRL 扩展默认值

CRL 签名证书，关于撤销证书
Request （请求），通过控制台请求证书

cRLDistributionPoints，CRLDistributionPoints
CRLNumber，CRLNumber
CRLReason，CRLReason
CRLs
何时进行自动更新，关于撤销证书

发布，关于撤销证书

发布到 LDAP 目录，发布 CRL，LDAP 发布所需的模式，配置 LDAP 目录

发布到文件，发布到文件

发布或发布点，CRL 颁发点

已定义，关于撤销证书

扩展，标准 X.509 v3 CRL 扩展参考

支持的扩展，关于撤销证书

特定于扩展的模块，关于 CRL 扩展

生成的时间，关于撤销证书

谁生成它，关于撤销证书

输入多次更新时间，为每个颁发点配置 CRL

输入更新周期，为每个颁发点配置 CRL

D

deltaCRLIndicator，deltaCRLIndicator

DER 编码的文件

查看内容，查看证书和 CRL 发布到文件

DN 组件映射器，LdapDNCompsMap

E

ECC

Request （请求），创建证书签名请求

配置，为证书设置签名算法

extensions，在 CA 证书中设置限制，证书和 CRL 的默认值、约束和扩展

authorityInfoAccess，authorityInfoAccess

authorityKeyIdentifier，在 CA 证书中设置限制，authorityKeyIdentifier，authorityKeyIdentifier

basicConstraints，basicConstraints

CA 证书和，在 CA 证书中设置限制

certificateIssuer，certificateIssuer

certificatePolicies，certificatePoliciesExt

cRLDistributionPoints，CRLDistributionPoints

CRLNumber，CRLNumber

CRLReason，CRLReason

deltaCRLIndicator，deltaCRLIndicator

extKeyUsage，extKeyUsage

invalidityDate，invalidityDate

issuerAltName，issuerAltName 扩展，issuerAltName

issuingDistributionPoint，issuingDistributionPoint

keyUsage，keyUsage

nameConstraints，nameConstraints

netscape-cert-type，netscape-cert-type

Netscape-defined，Netscape-Defined 证书扩展参考

policyConstraints，policyConstraints

policyMappings，policyMappings

privateKeyUsagePeriod，privateKeyUsagePeriod

subjectAltName，subjectAltName

subjectDirectoryAttributes，subjectDirectoryAttributes

X.509 CRL，总结，标准 X.509 v3 CRL 扩展参考

X.509 证书，总结，标准 X.509 v3 证书扩展参考

一个示例，标准 X.509 v3 证书扩展参考

生成工具，请求签名证书，请求其他证书

用于加入的工具，请求签名证书，请求其他证书

extKeyUsage，extKeyUsage

F

Federal Bridge 证书颁发机构，使用跨证书

G

groups

更改成员，更改组中的成员

I

invalidityDate，invalidityDate

IPv6

和 SCEP 证书，为路由器生成 SCEP 证书

issuerAltName，issuerAltName 扩展，issuerAltName

issuingDistributionPoint，issuingDistributionPoint

J

jobs

与插件实现相比，关于自动化作业

内置模块

unpublishExpiredCerts，unpublishExpiredCerts (UnpublishExpiredJob)

打开调度程序，设置作业调度程序

指定调度，自动任务的频率设置

设置频率，设置作业调度程序

配置作业通知消息，自定义 CA 通知消息，设置自动化作业

K

keyUsage，keyUsage

KRA 传输证书

Request （请求），通过控制台请求证书

L

LDAP 发布

已定义，LDAP 发布

手动更新，更新目录中的证书和 CRL

何时执行，手动更新目录中的证书

谁可以执行此操作，更新目录中的证书和 CRL

logging

buffered vs. unbuffered，缓冲和不缓冲的日志记录

从证书系统控制台管理，在控制台中查看日志

已记录的服务，正在记录的服务

日志文件

归档轮转文件，日志文件轮转

签名轮转的文件，签名日志文件

轮转的时间，日志文件轮转

默认位置，配置子系统日志

日志类型，配置子系统日志
Error，Tomcat 错误和访问日志

日志级别，日志级别(Message Categories)
它们如何与消息类别相关，日志级别(Message Categories)
选择正确的级别的权利，日志级别(Message Categories)
默认选择，日志级别(Message Categories)

M

Mappers
在安装过程中创建，创建映射程序，LdapCaSimpleMap，LdapSimpleMap

N

nameConstraints，nameConstraints
netscape-cert-type，netscape-cert-type
nickname
对于 CA 签名证书，CA 签名密钥和证书
对于 OCSP 签名证书，OCSP 签名密钥和证书
对于 SSL 服务器证书，SSL 服务器密钥检测和证书，SSL 服务器密钥检测和证书
对于 TLS 签名证书，OCSP 签名密钥和证书
对于子系统证书，子系统证书，子系统证书，子系统证书
用于签名证书，OCSP 签名密钥和证书
notifications
代理有关未发布的证书，unpublishExpiredCerts (UnpublishExpiredJob)
配置邮件服务器
hostname，为证书系统通知配置邮件服务器
端口，为证书系统通知配置邮件服务器

O

OCSP publisher，OCSPPublisher
OCSP 签名证书，OCSP 签名密钥和证书
nickname，OCSP 签名密钥和证书
Request （请求），通过控制台请求证书

P

PIN Generator 工具
向用户提供 PIN，设置基于 PIN 的注册

policyConstraints，policyConstraints

policyMappings，policyMappings

ports
用于通知的邮件服务器，为证书系统通知配置邮件服务器

privateKeyUsagePeriod，privateKeyUsagePeriod

profiles
配置集的工作方式，Enrollment Profile

publishers
在安装过程中创建，配置 LDAP 站，LdapCaCertPublisher，LdapUserCertPublisher，LdapCertificatePairPublisher

R

restore，备份和恢复实例目录

roles
agent，代理

RSA
配置，为证书设置签名算法

S

SCEP
使用单独的身份验证证书，为 SCEP 配置安全设置
启用，启用 SCEP 注册
设置允许的算法，为 SCEP 配置安全设置
设置非大小，为 SCEP 配置安全设置

SCEP 证书
和 IPv6，为路由器生成 SCEP 证书

SMTP 设置，为证书系统通知配置邮件服务器

SSL 客户端证书
Request （请求），通过控制台请求证书

SSL 服务器证书，SSL 服务器密钥检测和证书，SSL 服务器密钥检测和证书
nickname，SSL 服务器密钥检测和证书，SSL 服务器密钥检测和证书
Request （请求），通过控制台请求证书
删除，从数据库中删除证书
更改信任设置，更改 CA 证书的信任设置
查看详情，通过控制台查看数据库内容

Status 标签页，将 pkiconsole 用于 CA、OCSP、KRA 和 TKS 子系统
subjectAltName，subjectAltName
subjectDirectoryAttributes，subjectDirectoryAttributes
subjectKeyIdentifier
subjectKeyIdentifier，subjectKeyIdentifier

sudo
管理员的权限，为证书系统服务设置 sudo 权限

T

templates
对于通知，自定义 CA 通知消息

TLS CA 签名证书，OCSP 签名密钥和证书
nickname，OCSP 签名密钥和证书

TPS
users，为 TPS 创建和管理用户
设置配置集，为用户设置配置集

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 Red Hat Certificate System 的版本号。

修订 10.1-1 **Mon Jan 25 2021** **Florian Delehaye**

各种修复及改进。

修订 10.1-0 **Wed Dec 02 2020** **Florian Delehaye**

本指南的 Red Hat Certificate System 10.1 发行版。

修订 10.0-0 **Wed Sep 17 2020** **Florian Delehaye**

本指南的 Red Hat Certificate System 10.0 发行版本。