Red Hat Certificate System 9

管理ガイド (Common criteria Edition)

Red Hat Certificate System 9.4 Common Criteria 認定の更新
エディッション 9.4-1

Last Updated: 2023-04-20

Red Hat Certificate System 9 管理ガイド (Common criteria Edition)

Red Hat Certificate System 9.4 Common Criteria 認定の更新
エディッション 9.4-1

Red Hat
Customer Content Services

法律上の通知

概要

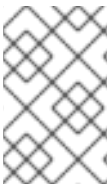
本ガイドでは、Common Criteria 環境で、Certificate System 9.4 のインストール、設定、および管理のすべての側面を説明します。また、ユーザーの追加、証明書の要求、更新、および失効、CRL の公開、スマートカードの管理などの管理タスクも説明します。本ガイドは、Certificate System の管理者を対象としています。

目次

第1章 RED HAT CERTIFICATE SYSTEM サブシステムの概要

注記

この章では、Red Hat Certificate System とさまざまなサブシステムの概要について説明します。評価済みの製品機能の詳細は、の NIAP Product Compliant List https://www.niap-ccevs.org/Product/PCL.cfm を参照してください。



注記

暗号プロバイダーとして評価されたのは、NSS (Network Security Services) と FIPS(Federal Information Processing Standard) の HSM(Hardware Security Module) のみであった。

すべての一般的な PKI 操作 (証明書の発行、更新、取り消しなど、キーのアーカイブと回復、CRL の公開と証明書ステータスの検証) は、Red Hat Certificate System 内の相互運用サブシステムによって実行されます。この章では、個々のサブシステムの機能と、それらが連携して堅牢でローカルな PKI を確立する方法を説明します。

1.1. 証明書に使用

証明書の目的は、信頼を確立することです。その使用法は、それが保証するために使用される信頼の種類によって異なります。提示した者の ID を確認するために、いくつかの種類の証明書が使用されたり、オブジェクトまたはアイテムが改ざんされていないことを確認したりするために使用されます。

証明書の使用方法、証明書の種類、または証明書が ID や関係を確立する方法については、『Red Hat 証明書システム計画、インストール、および展開ガイド (Common Criteria Edition)』) の 『証明書および認証』 のセクションを参照してください。

1.2. CERTIFICATE SYSTEM サブシステムのレビュー

Red Hat Certificate System は 5 つの異なるサブシステムを提供します。それぞれは、PKI デプロイメントのさまざまな側面に重点を置いています。これらのサブシステムは連携して、公開鍵インフラストラクチャー (PKI) を作成します。インストールされているサブシステムに応じて、PKI はトークン管理システム (TMS) または非トークン管理システムとして機能できます。サブシステムおよび TMS と非 TMS 環境の説明については、『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』) の 『A Review ofCertificate System』 Subsystems を参照してください。

1.3. 証明書管理の概観 (非 TMS)

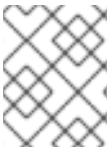
従来の PKI 環境は、ソフトウェアデータベースに保存されている証明書を管理する基本的なフレームワークを提供します。これは、スマートカードで証明書を管理しないため、**TMS 以外** の環境です。少なくとも、TMS 以外の環境では CA のみが必要ですが、TMS 以外の環境では OCSP レスポンダー と KRA インスタンスも使用できます。

このトピックについては、『Red Hat 証明書システム計画、インストール、および展開ガイド (Common Criteria Edition)』 の以下のセクションを参照してください。

- 『証明書の管理』
- 『単一 Certificate Manager の使用』

- 『紛失したキーの計画: キーのアーカイブと回復』
- 『証明書要求の処理の分散』
- 『クライアント OCSP 要求の分散』

1.4. TOKEN MANAGEMENT SYSTEM (TMS) の概観



注記

TMS 上の本セクションにある機能は、評価でテストされていません。本セクションは参照用途としてのみ提供されています。

証明書システムは、証明書の作成、管理、更新、取り消しを行い、鍵のアーカイブおよび復元も行います。スマートカードを使用する組織の場合は、Certificate System に、トークン管理システムがあります。これは、鍵と要求を生成し、スマートカードに使用される証明書を受け取るために、確立された関係を持つサブシステムのコレクションです。

このトピックについては、『Red Hat 証明書システム計画、インストール、および展開ガイド (Common Criteria Edition)』 の以下のセクションを参照してください。

- 『スマートカードとの連携 (TMS)』
- 『スマートカードの使用』

1.5. RED HAT CERTIFICATE SYSTEM サービス

証明書やサブシステムを管理するためのインターフェイスは、管理者、エージェント、監査人、エンドユーザーなど、ユーザーのロールに応じてさまざまな種類があります。各インターフェイスで実行されるさまざまな機能の概要については、『Red Hat Certificate System 9 Planning、Installation、and Deployment Guide (Common Criteria Edition』) の 『Red Hat Certificate System 』 User Interfaces のセクションを参照してください。

パート I. RED HAT CERTIFICATE SYSTEM ユーザーインターフェイス

第2章 ユーザーインターフェイス

ユーザーロール (管理者、エージェント、監査ユーザー、エンドユーザー) に応じて、証明書やサブシステムの管理にはさまざまなインターフェイスがあります。

2.1. ユーザーインターフェイスの概要

管理者は、以下のインターフェイスを使用して、完全な Certificate System インストールと安全に対話できます。

- PKI コマンドラインインターフェイスおよびその他のコマンドラインユーティリティー
- PKI コンソールのグラフィカルインターフェイス
- Certificate System Web インターフェイス

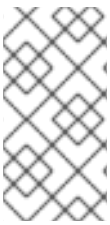
これらのインターフェイスには、TLS による Certificate System サーバーとの安全な通信に使用する前に設定が必要です。適切な設定なしでこれらのクライアントを使用することはできません。これらのツールの一部は、TLS クライアント認証を使用します。必要に応じて、必要な初期化手順にこれの設定が含まれます。使用するインターフェイスは、管理者の設定と機能によって異なります。これらのインターフェイスを使用する一般的なアクションは、本章の後半の残りのガイドに記載されています。

デフォルトでは、PKI コマンドラインインターフェイスは、ユーザーの ~/**.dogtag/nssdb**/ ディレクトリーにある NSS データベースを使用します。「pki CLI の初期化」では、NSS データベースを管理者の証明書およびキーで初期化する詳細な手順を説明します。PKI コマンドラインユーティリティーの使用例は、「pki CLI の使用」に記載されています。その他の例を以下に示します。

(他のユーザーロールの管理者として) 証明書システムとの干渉は、さまざまなコマンドラインユーティリティーを使用して、CMC 要求の送信、生成された証明書の管理などを行うことができます。これらについては、「AtoB」など、「コマンドラインインターフェイス」で簡単に説明します。これらのユーティリティーは、「**PKCS10Client** を使用した CSR の作成」 などの後のセクションで使用されています。

Certificate System の PKI コンソールインターフェイスは、グラフィカルインターフェイスです。「**pkiconsole** の初期化」 は、初期化方法を説明します。「CA、OCSP、KRA、および TKS サブシステムに対する **pkiconsole** の使用」 では、コンソールインターフェイスの使用の概要を説明します。「Java ベースの管理コンソールを使用した証明書の登録プロファイルの管理」 などの後のセクションでは、特定の操作について詳しく説明します。

Certificate System Web インターフェイスを使用すると、Firefox Web ブラウザーから管理アクセスが可能になります。「ブラウザーの初期化」 は、クライアント認証の設定手順を説明します。「Web インターフェイス」 の他のセクションでは、証明書システムの Web インターフェイスの使用を説明します。特定のタスクのためのブラウザーの使用に関する詳細な情報は、「証明書の検索 (詳細)」 などの他のドキュメントに含まれています。



注記

PKI コンソールセッションを終了するには、**Exit** (終了) ボタンをクリックします。Web ブラウザーセッションを終了するには、ブラウザーを閉じます。コマンドラインユーティリティーは、アクションを実行してプロンプトに返すとすぐにそれ自身を終了するため、セッションを終了するには、管理者の部分でアクションは必要ありません。

2.2. クライアント NSS データベースの初期化

Red Hat Certificate System では、特定のインターフェイスが TLS クライアント証明書認証 (通常は認証) を使用してサーバーにアクセスしないといけない場合があります。サーバー側の管理タスクを実行する前に、以下を行う必要があります。

1. クライアント用の NSS データベースを準備します。これは、新規データベースか、または既存のデータベースにすることができます。
2. CA 証明書チェーンをインポートして信頼します。
3. 証明書と対応するキーがあります。NSS データベースで生成したり、PKCS #12 ファイルから他の場所からインポートしたりできます。

ユーティリティーに基づいて、NSS データベースを適宜初期化する必要があります。以下を参照してください。

- 「pki CLI の初期化」
- 「**pkiconsole** の初期化」
- 「ブラウザーの初期化」

2.3. グラフィカルインターフェイス

pkiconsole は、Administrator ロール権限を持つユーザー向けにサブシステム自体を管理するグラフィカルインターフェイスです。これには、ユーザーの追加、ログの設定、プロファイルおよびプラグインの管理、および内部データベースなどの多くの機能が含まれます。このユーティリティーは、クライアント認証を使用して TLS 経由で Certificate System サーバーと通信し、リモートでサーバーを管理するために使用できます。

2.3.1. pkiconsole の初期化

pkiconsole インターフェイスを初めて使用するには、新しいパスワードを指定し、以下のコマンドを使用します。

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

このコマンドは、~/**.redhat-idm-console**/ ディレクトリーに新しいクライアントの NSS データベースを作成します。

CA 証明書を PKI クライアント NSS データベースにインポートするには、「ルート証明書のインポート」 を参照してください。

新しいクライアント証明書を要求するには、5章*証明書の要求、登録、および管理*を参照してください。

以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

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

11章*証明書/キー暗号トークンの管理* の説明に従って、管理クライアント証明書を検証し、インポートします。

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

重要

CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

既存のクライアント証明書とそのキーをクライアント NSS データベースにインポートするには、次を実行します。

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

以下のコマンドを使用して、クライアント証明書を確認します。

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

2.3.2. CA、OCSP、KRA、および TKS サブシステムに対する pkiconsole の使用

Java コンソールは、CA、OCSP、KRA、および TKS の 4 つのサブシステムで使用されます。コンソールには、ローカルにインストールされた **pkiconsole** ユーティリティーを使用してアクセスできます。コマンドにはホスト名、サブシステムの管理 TLS ポート、特定のサブシステムタイプが必要なため、あらゆるサブシステムにアクセスできます。

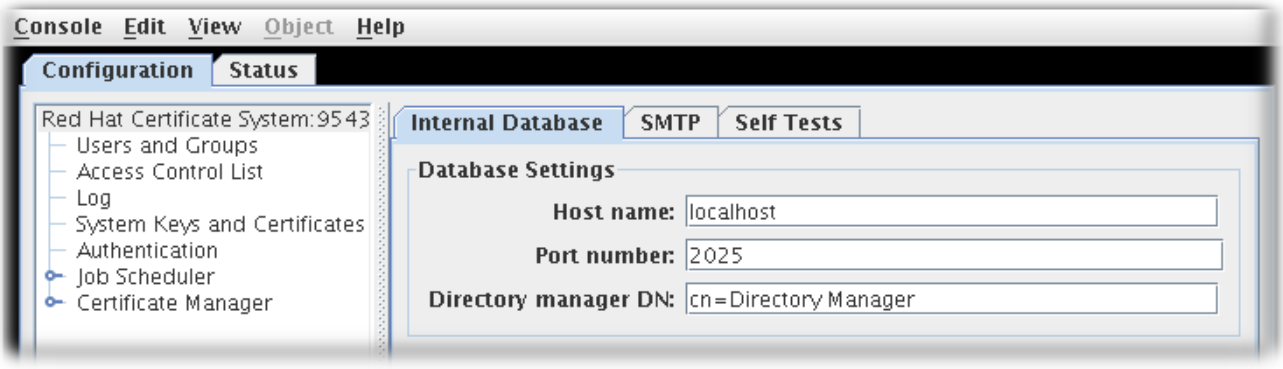
```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

DNS が設定されていない場合は、IPv4 アドレスまたは IPv6 アドレスを使用して、コンソールに接続できます。たとえば、以下のようになります。

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

これにより、図2.1「Certificate System コンソール」 にあるようにコンソールが開きます。

図2.1 Certificate System コンソール



Configuration タブは、名前が示すように、サブシステムのすべての設定を制御します。このセクションで利用可能な選択肢は、インスタンスがどのサブシステムタイプであるかによって異なります。CA にはジョブ、通知、および証明書登録認証の追加設定があるため、CA にはほとんどのオプションがあります。

すべてのサブシステムには 4 つの基本的なオプションがあります。

- ユーザーおよびグループ

- アクセス制御リスト
- ログ設定
- サブシステム証明書 (セキュリティードメインや監査署名など、サブシステムに発行した証明書)

Status タブには、サブシステムによってメンテナーンスされるログが表示されます。

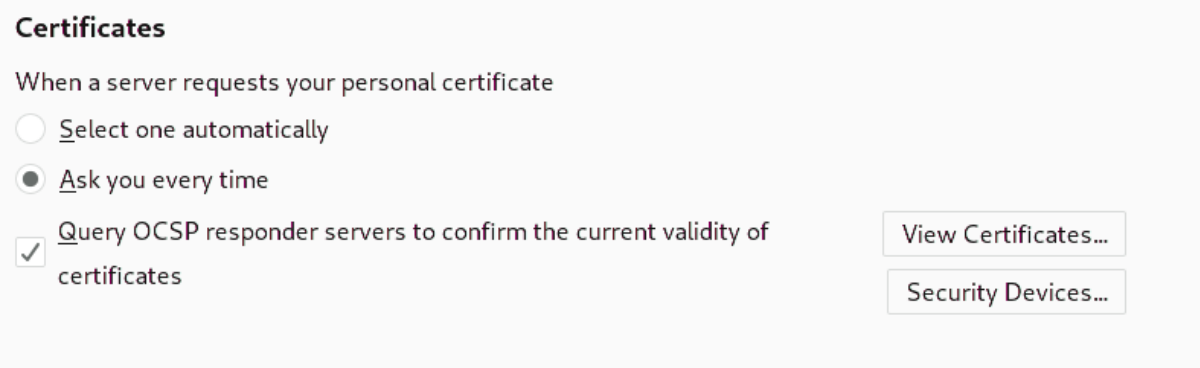
2.4. WEB インターフェイス

2.4.1. ブラウザーの初期化

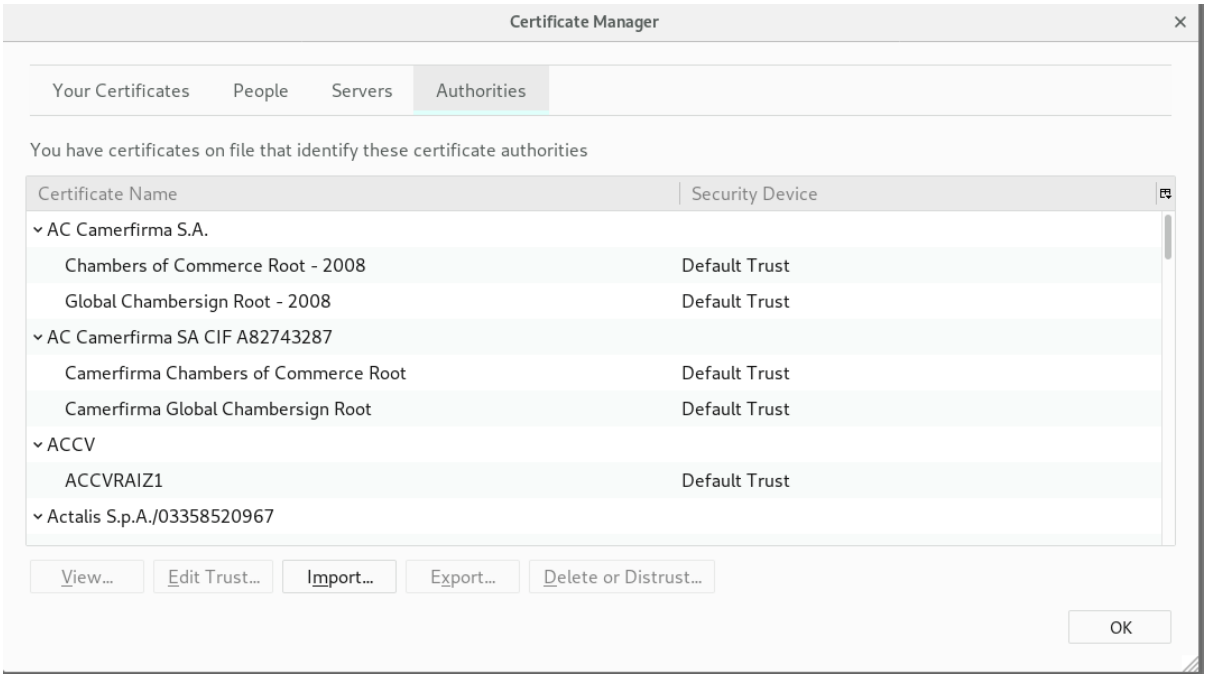
本セクションでは、Firefox が PKI サービスにアクセスするためのブラウザーの初期化を説明します。

CA 証明書のインポート

1. **Menu → Preferences → Privacy & Security→ View certificates**をクリックします。



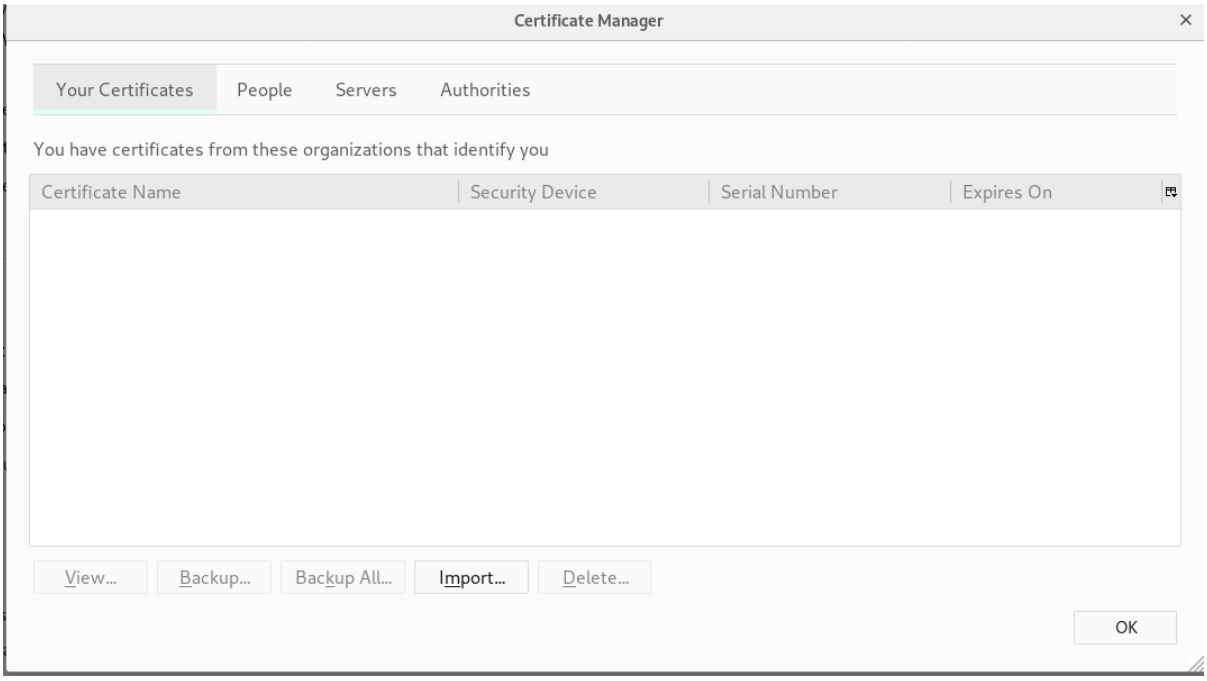
2. **Authorities** タブを選択し、**Import** ボタンをクリックします。



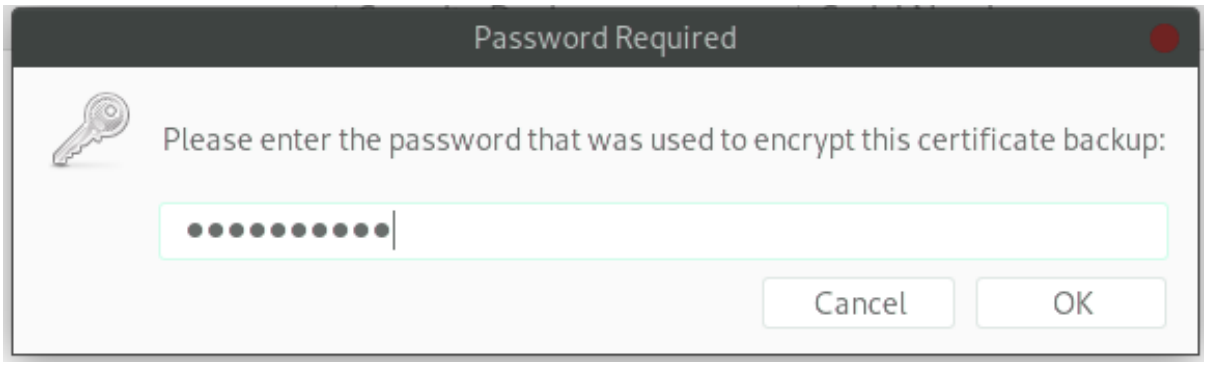
3. **ca.crt** ファイルを選択して、**Import** をクリックします。

クライアント証明書のインポート

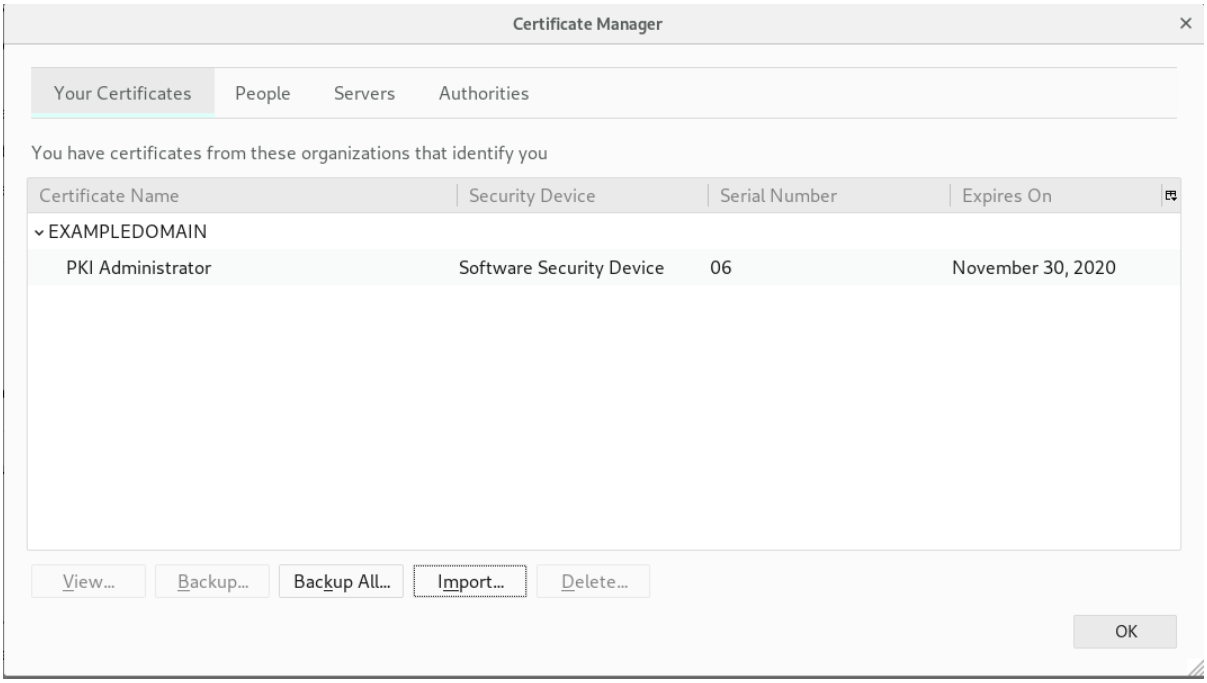
1. **Options → Preferences → Privacy & Security → View certificates** をクリックします。
2. **Your Certificates** タブを選択します。



3. **Import** をクリックして、**ca_admin_cert.p12** などのクライアント p12 ファイルを選択します。
4. プロンプトにクライアント証明書のパスワードを入力します。



5. **OK** をクリックします。
6. **Your Certificates** の下にエントリーが追加されていることを確認します。



Web コンソールへのアクセス
ブラウザーで **https://*host_name*:*ポート*** を開いて PKI サービスにアクセスできます。

2.4.2. 管理インターフェイス

すべてのサブシステムは HTML ベースの管理インターフェイスを使用します。ホスト名を入力し、URL としてセキュアなポートを入力し、管理者の証明書で認証し、適切な **Administrators** リンクをクリックします。

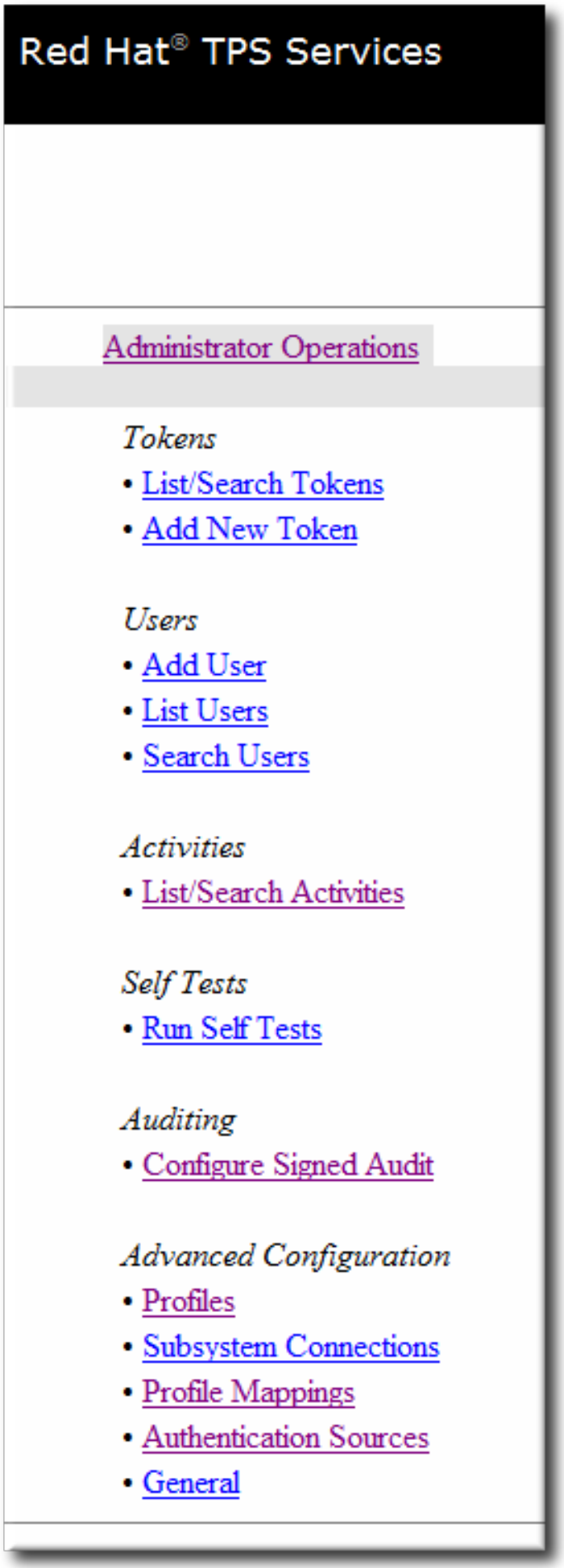
注記

管理者およびエージェントサービスの両方に使用されるすべてのサブシステムには、1 つの TLS ポートがあります。これらのサービスへのアクセスは、証明書ベースの認証により制限されます。

HTML 管理インターフェイスは Java コンソールよりもはるかに制限されています。プライマリー管理機能はサブシステムユーザーを管理します。

TPS では、操作は TPS サブシステムのユーザー管理のみを許可します。ただし、TPS 管理ページでは、トークンを一覧表示し、TPS で実行しているすべてのアクティビティー (通常は非表示管理アクションを含む) をすべて表示できます。

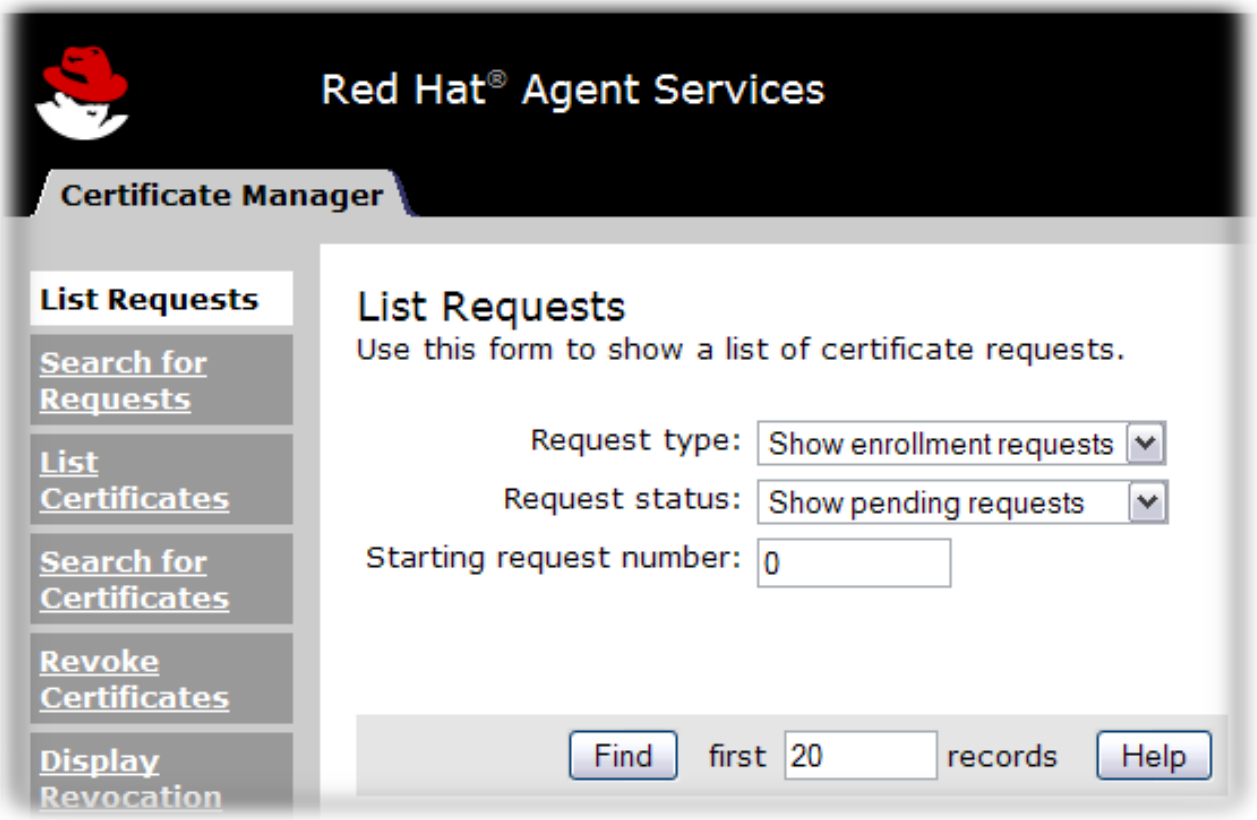
図2.2 TPS 管理ページ


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. エージェントインターフェイス

エージェントサービスページは、証明書およびトークン管理タスクがほぼすべて実行されます。これらのサービスは HTML ベースのもので、エージェントは特別なエージェント証明書を使用してサイトに対して認証されます。

図2.3 Certificate Manager のエージェントサービスページ



操作はサブシステムによって異なります。

- Certificate Manager エージェントサービスには、(証明書を発行する) 証明書要求の承認、証明書の失効、ならびに証明書および CRL の公開が含まれます。CA が発行するすべての証明書は、そのエージェントサービスページで管理できます。
- TPS エージェントサービスは、CA エージェントサービスと同様、フォーマットされ、TPS を介して証明書が発行されたすべてのトークンを管理します。トークンはエージェントで登録、一時停止、および削除できます。他の 2 つのロール (operator および admin) は Web サービスページでトークンを表示できますが、トークンに対するアクションを実行できません。
- KRA エージェントサービスページは、キーリカバリー要求を処理します。キーリカバリー要求は、証明書が失われた場合に既存のキーペアを再利用して証明書を発行できるようにするかどうかを設定します。
- OCSP エージェントサービスページを使用すると、エージェントは CRL を OCSP に公開し、CRL を手動で OCSP に読み込み、クライアント OCSP 要求の状態を表示するように CA を設定します。

TKS は、エージェントサービスページのない唯一のサブシステムです。

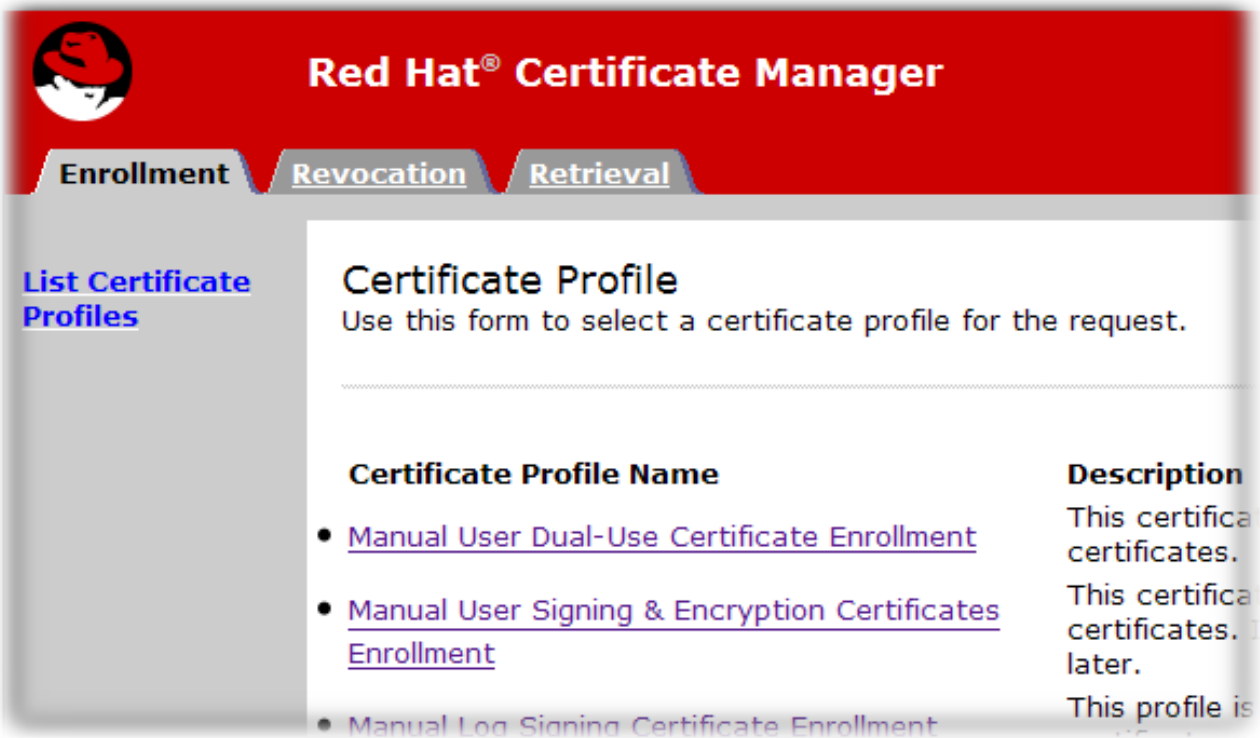
2.4.4. エンドユーザーページ

CA と TPS の両方は、ある方法で直接ユーザー要求を処理します。つまり、エンドユーザーにはこれらのサブシステムに接続する方法が必要です。CA にはエンドユーザーまたは **エンドエンティティー** の HTML サービスがあります。TPS は、Enterprise Security Client を使用します。

エンドユーザーサービスは、サーバーのホスト名と標準のポート番号を使用して標準の HTTP 経由でアクセスします。また、サーバーのホスト名および特定のエンドエンティティー TLS ポートを使用して、HTTPS 経由でもアクセスできます。

CA の場合、各タイプの TLS 証明書は、**プロファイル** と呼ばれる特定のオンライン送信フォームで処理されます。CA には約 20 ダースの証明書プロファイルがあり、証明書の種類 (ユーザー TLS 証明書、サーバー TLS 証明書、ログおよびファイル署名証明書、電子メール証明書、電子メール証明書、およびあらゆる種類のサブシステム証明書) に対応しています。カスタムプロファイルもあります。

図2.4 Certificate Manager のエンドエンティティーページ



エンドユーザーは、証明書の発行時に CA ページから証明書を取得します。また、CA チェーンと CRL をダウンロードし、それらのページから証明書を取り消したり更新したりすることもできます。

2.5. コマンドラインインターフェイス

本セクションでは、コマンドラインユーティリティーを説明します。

2.5.1. pkiCLI

pki コマンドラインインターフェイス(CLI)は、REST インターフェイスを使用してサーバー上のさまざまなサービスへのアクセスを提供します(『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)の 『REST』 インターフェイス セクションを参照してください)』。CLI は以下のように呼び出すことができます。

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

CLI オプションは、コマンドの前に配置する必要があり、コマンドの後にコマンドパラメーターを指定する必要があることに注意してください。

2.5.1.1. pki CLI の初期化

コマンドラインインターフェイスを初めて使用するには、新しいパスワードを指定し、以下のコマンドを使用します。

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

これにより、~/**.dogtag/nssdb** ディレクトリーに新しいクライアント NSS データベースが作成されます。パスワードは、クライアントの NSS データベースを使用するすべての CLI 操作に指定する必要があります。または、パスワードがファイルに保存されている場合は、**-C** オプションを使用してファイルを指定できます。以下に例を示します。

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

CA 証明書をクライアント NSS データベースにインポートするには、「ルート証明書のインポート」を参照してください。

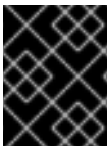
コマンドによっては、クライアント証明書の認証が必要な場合があります。既存のクライアント証明書とその鍵をクライアント NSS データベースにインポートするには、PKCS #12 ファイルとパスワードを指定して、以下のコマンドを実行します。

以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

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

11章 *証明書/キー暗号トークンの管理* の説明に従って、管理クライアント証明書を検証し、インポートします。

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



重要

CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

既存のクライアント証明書とその鍵をクライアント NSS データベースにインポートするには、PKCS #12 ファイルとパスワードを指定して、以下のコマンドを実行します。

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

以下のコマンドを使用して、クライアント証明書を確認します。

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

2.5.1.2. pki CLI の使用

コマンドラインインターフェイスは、階層構造で多数のコマンドをサポートします。トップレベルのコマンドを一覧表示するには、追加のコマンドまたはパラメーターを指定せずに **pki** コマンドを実行します。

```
$ pki
```

コマンドにはサブコマンドがあります。一覧を表示するには、コマンド名を指定して **pki** を実行して追加のオプションを指定せずに実行します。以下に例を示します。

```
$ pki ca
```

```
$ pki ca-cert
```

コマンドの使用情報を表示するには **--help** オプションを使用します。

```
$ pki --help
```

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

man ページを表示するには、コマンドラインの **help** コマンドを指定します。

```
$ pki help
```

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

認証を必要としないコマンドを実行するには、コマンドとそのパラメーター (必要な場合) を指定します。以下に例を示します。

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

クライアント証明書の認証を必要とするコマンドを実行するには、証明書のニックネーム、クライアント NSS データベースのパスワード、および任意のサーバーの URL を指定します。

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

以下に例を示します。

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

デフォルトでは、CLI は **http://*local_host_name*:8080** でサーバーと通信します。別の場所でサーバーと通信するには、URL を **-U** オプションで指定します。以下に例を示します。

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

2.5.2. AtoB

AtoB ユーティリティーは、Base64 でエンコードされた証明書を、同等のバイナリーにデコードします。以下に例を示します。

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

詳細情報、その他のオプション、およびその他の例は、AtoB(1) の man ページを参照してください。

2.5.3. AuditVerify

AuditVerify ユーティリティーは、ログエントリーの署名を検証して、監査ログの整合性を検証します。

たとえば、以下のようになります。

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

この例では、**~jsmith/auditVerifyDir** NSS データベー (**-d**) の **Log Signing Certificate** (**-n**) を使用して監査ログを検証します。検証するログのリスト (**-a**) は **~jsmith/auditVerifyDir/logListFile** ファイルにあります。こちらは、コンマ区切りで時系列順に並べられています。証明書の先頭に接頭辞 (**-P**) を追加し、キーデータベースのファイル名を空にします。出力は詳細です (**-v**)。

詳細、より多くのオプション、および追加の例については、AuditVerify(1) の man ページまたは 14.2.2 を参照してください。Red Hat Certificate System 管理ガイドの署名付き監査ログの使用。

2.5.4. BtoA

BtoA ユーティリティーは、バイナリーデータを Base64 でエンコードします。以下に例を示します。

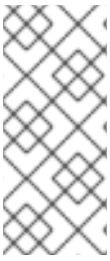
```
$ BtoA input.bin output.ascii
```

詳細情報、その他のオプション、およびその他の例は、BtoA(1) の man ページを参照してください。

2.5.5. CMCRequest

CMCRequest ユーティリティーは、証明書の発行または失効要求を作成します。以下に例を示します。

```
$ CMCRequest example.cfg
```



注記

CMCRequest ユーティリティーのすべてのオプションは、ユーティリティーに渡される設定ファイルの一部として指定されます。設定ファイルのオプションと詳細は、CMCRequest(1) の man ページを参照してください。4.3 も参照してください。CMC および 6.2.1 を使用した証明書の要求と受け取りRed Hat Certificate System 管理ガイドの CMCRequest を使用した証明書の取り消し

2.5.6. CMCRevoke

レガシー。使用しないでください。

2.5.7. CMCSharedToken

CMCSharedToken ユーティリティーは、共有秘密の CMC リクエストのユーザーパスフレーズを暗号化します。以下に例を示します。

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

共有パスフレーズ (**-s**) は、現在のディレクトリー (**-d**) にある NSS データベースにある **subsystemCert cert-pki-tomcat** (**-n**) という名前の証明書を使用して、**cmcSharedtok2.b64** ファイル (**-o**) に暗号化されて保存されます。デフォルトのセキュリティートークン **internal** が使用され (**-h** が指

定されていないため)、トークンへのアクセスには **myNSSPassword** のトークンパスワードが使用されます。

詳細、その他のオプション、およびその他の例は、CMCSharedtoken(1) の man ページおよび 8.1.3.1 「Red Hat Certificate System 管理ガイドの共有シークレットトークンの作成」を参照してください。

2.5.8. CRMFPopClient

CRMFPopClient ユーティリティーは、NSS データベースを使用する Certificate Request Message Format (CRMF) クライアントであり、Possession の Proof を提供します。

たとえば、以下のようになります。

```
$ 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
```

この例では、**cn=subject_name** サブジェクト DN (**-n**)、現在のディレクトリー (**-d**) の NSS データベース (-b)、トランスポート (**kra.transport**) (**-b**) に使用する証明書、**AES/CBC/PKCS5Padding** キーをラップする証明書で新しい CSR を作成します (**-v**)。また、生成される CSR が **/user_or_entity_database_directory/example.csr** ファイル (**-o**) に書き込まれます。

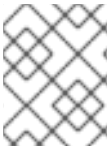
詳細情報、その他のオプション、およびその他の例は、**CRMFPopClient --help** コマンドの出力と 4.2.4 も参照してください。「Red Hat Certificate System 管理ガイドの CRMFPopClient を使用した CSR」の作成を参照してください。

2.5.9. HttpClient

HttpClient ユーティリティーは、CMC 要求を送信するための NSS 対応の HTTP クライアントです。

たとえば、以下のようになります。

```
$ HttpClient request.cfg
```



注記

HttpClient ユーティリティーへのすべてのパラメーターは **request.cfg** ファイルに保存されます。詳細は、**HttpClient --help** コマンドの出力を参照してください。

2.5.10. OCSPClient

証明書失効リストのステータスを確認する Online Certificate Status Protocol (OCSP) クライアント。

たとえば、以下のようになります。

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

この例では、ポート **8080** (**-p**) の **server.example.com** OCSP サーバー (**-h**) に対してクエリーを実行し、シリアル番号 **2** (**--serial**) を持つ **caSigningcet cert-pki-ca** (**-c**) が署名した証明書を確認します。**/etc/pki/pki-tomcat/alias** ディレクトリーの NSS データベースが使用されます。

詳細情報、その他のオプション、およびその他の例は、**OCSPClient --help** コマンドの出力を参照してください。

2.5.11. PKCS10Client

PKCS10Client ユーティリティーは、必要に応じて HSM 上に RSA キーおよび EC キーの CSR を PKCS10 形式で作成します。

たとえば、以下のようになります。

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

この例では、**/etc/dirsrv/slapd-instance_name/** ディレクトリー (**-d**) に 2048 ビット (**-l**) で、新しい RSA (**-a**) キーを、データベースパスワード (**password**) (**-p**) で作成します。出力 CSR は **~/ds.cfg** ファイル (**-o**) に格納されます。また、証明書 DN は **CN=$HOSTNAME** (**-n**) です。

詳細情報、その他のオプション、およびその他の例は、PKCS10Client(1) の man ページを参照してください。

2.5.12. PrettyPrintCert

PrettyPrintCert ユーティリティーは、人間が判読可能な形式で証明書の内容を表示します。

たとえば、以下のようになります。

```
$ PrettyPrintCert ascii_data.cert
```

このコマンドは、**ascii_data.cert** ファイルの出力を解析し、その内容を人間が読める形式で表示します。出力には、署名アルゴリズム、指数、モジュール、証明書拡張などの情報が含まれます。

詳細情報、その他のオプション、およびその他の例は、PrettyPrintCert(1) の man ページを参照してください。

2.5.13. PrettyPrintCrl

PrettyPrintCrl ユーティリティーは、CRL ファイルの内容を人間が判読できる形式で表示します。

たとえば、以下のようになります。

```
$ PrettyPrintCrl ascii_data.crl
```

このコマンドは、**ascii_data.crt** ファイルの出力を解析して、その内容を人間が読める形式で表示します。出力には、失効署名アルゴリズム、失効の発行者、取り消された証明書とその理由が一覧になったものなどが含まれます。

詳細情報、その他のオプション、およびその他の例は、PrettyPrintCrl(1) の man ページを参照してください。

2.5.14. TokenInfo

TokenInfo ユーティリティーは、NSS データベース内のトークンを一覧表示します。

たとえば、以下のようになります。

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

このコマンドは、指定のデータベースディレクトリーに登録されたすべてのトークン (HSM、ソフトトークンなど) を表示します。

詳細情報、その他のオプション、およびその他の例は、**TokenInfo** コマンドの出力を参照してください。

2.5.15. tkstool

tkstool ユーティリティーは、トークンキーサービス (TKS) サブシステムと対話します。

たとえば、以下のようになります。

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

このコマンドは、HSM **token_name** の **/var/lib/pki/pki-tomcat/alias** NSS データベースに **new_master** (**-n**) という名前の新しいマスターキー (**-M**) を作成します。

詳細情報、その他のオプション、およびその他の例は、**tkstool -H** コマンドの出力を参照してください。

パート II. 証明書サービスの設定

注記

インストール後に **CS.cfg**、**server.xml**、または設定ファイルを変更すると、認定環境で明示的に禁止されます。

第3章 証明書を発行するルール (証明書プロファイル) の作成

Certificate System は、受信証明書要求にポリシーを適用し、入力要求タイプと出力証明書タイプを制御するためのカスタマイズ可能なフレームワークを提供します。これらは **証明書プロファイル** と呼ばれます。証明書プロファイルは、Certificate Manager のエンドエンティティーページで証明書登録フォームに必要な情報を設定します。本章では、証明書プロファイルの設定方法を説明します。

3.1. 証明書プロファイルの概要

証明書プロファイルは、認証方法、認可方法、証明書のデフォルトの内容、内容の値の制約、証明書プロファイルの入力と出力の内容など、特定の種類の証明書の発行に関連するすべてを定義します。登録要求および更新要求は証明書プロファイルに送信され、その証明書プロファイルで設定されたデフォルトと制約の対象となります。これらの制約は、要求が証明書プロファイルに関連付けられた入力フォームを介して送信されるか、他の手段を介して送信されるかに関係なく適用されます。証明書プロファイル要求から発行される証明書には、デフォルトで必要なコンテンツと、デフォルトのパラメーターで必要な情報が含まれています。制約は、証明書で許可されるコンテンツに対するルールを提供します。

証明書プロファイルの使用およびカスタマイズの詳細は、「証明書プロファイルの設定」 を参照してください。

Certificate System には、デフォルトのプロファイルのセットが含まれています。デフォルトのプロファイルは、ほとんどのデプロイメントを満たすために作成されますが、すべてのデプロイメントは独自の新規証明書プロファイルを追加するか、既存のプロファイルを変更することができます。

- **認証。** すべての認証プロファイルで認証方法を指定できます。
- **認可。** すべての認定プロファイルで承認方法を指定できます。
- **プロファイル入力。** プロファイルの入力は、証明書が要求されたときに CA に送信されるパラメーターおよび値です。プロファイル入力には、証明書要求の公開鍵と、証明書の終了エンティティーによって要求される証明書のサブジェクト名が含まれます。
- **プロファイルの出力。** プロファイルの出力は、エンドエンティティーに証明書を提供する形式を指定するパラメーターおよび値です。プロファイルの出力は、要求が成功したときに PKCS#7 証明書チェーンが含まれる CMC の応答です。
- **証明書の内容。** 各証明書は、割り当てられたエンティティーの名前 (サブジェクト名)、署名アルゴリズム、有効期間などのコンテンツ情報を定義します。証明書に含まれるものは、X.509 標準で定義されます。X509 標準のバージョン 3 では、証明書に拡張機能を含めることもできます。証明書エクステンションの詳細は、??? を参照してください。

  証明書プロファイルに関する情報はすべて、プロファイルの設定ファイルのプロファイルポリシーの **set** エントリーで定義されます。複数の証明書が同時に要求される可能性がある場合は、各証明書のニーズを満たすためにプロファイルポリシーに複数のセットエントリーを定義できます。各ポリシーセットは多数のポリシールールで設定され、各ポリシールールは証明書コンテンツのフィールドを記述します。ポリシールールには、以下の内容を含めることができます。

  - **プロファイルのデフォルトです。** これらは、証明書内に含まれる情報に対する事前定義済みのパラメーターおよび許可される値です。プロファイルのデフォルトには、証明書の有効期間と、発行する証明書のタイプにどの証明書拡張機能が表示されるかが含まれます。
  - **プロファイルの制約。** 制約は、証明書を発行するルールまたはポリシーを設定します。また、プロファイル制約には、証明書のサブジェクト名に少なくとも 1 つの CN コンポーネントを含める必要があるルールが含まれます。また、証明書の有効性を最大 360 日に設定し

て、更新を許可する猶予期間を定義するルール、または **subjectaltname** 拡張が常に **true** に設定される必要があるというルールが含まれます。

3.1.1. 登録プロファイル

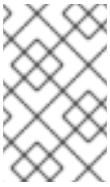
入力、出力、およびポリシーセットを定義する各プロファイルのパラメーターは、Table 11.1 に詳細に記載されています。Red Hat Certificate System 計画、インストールガイド、およびデプロイメントのガイドのプロファイル設定ファイルのパラメーター

プロファイルには、通常 例3.1「caCMCUserCert プロファイルの例」 の **caUserCert** プロファイルで説明されているように、入力、ポリシーセット、および出力が含まれます。

例3.1 caCMCUserCert プロファイルの例

証明書プロファイルの最初の部分は説明です。これは、名前、長い説明、有効かどうか、および有効であるかを表示します。

```
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
```



注記

このプロファイルにない **auth.instance_id=** エントリーは、このプロファイルを使用した登録リクエストの送信に認証は必要ありません。ただし、保証を取得するには、承認された CA エージェントによる手動承認が必要です。

次に、プロファイルはプロファイルに必要なすべての入力を一覧表示します。

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

caCMCUserCert プロファイルの場合、これは、証明書要求タイプ (CMC) を定義します。

次に、プロファイルは出力 (最終証明書の形式) を定義する必要があります。唯一利用できるのは **certOututIcmpl** で、成功すると、CMC 応答が要求元に戻ります。

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

最後の (最大の) 設定ブロックは、プロファイルに設定されたポリシーです。ポリシーは、有効期間、更新設定、証明書が使用できるアクションなど、最終的な証明書に適用されるすべての設定一覧を設定します。**policyset.list** パラメーターは、1 つの証明書に適用されるポリシーのブロック名を識別します。適用する個々のポリシーが **policyset.userCertSet.list** により一覧表示されます。

たとえば、6 番目のポリシーは、ポリシーの設定に従って、証明書に Key Usage Extension を自動的に入力します。これは、デフォルトを設定し、制約を設定して証明書でそれらのデフォルトを使用するようにする必要があります。

```
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. 証明書拡張: デフォルトおよび制約

拡張機能は、証明書の使用方法に関する証明書またはルールに含める追加情報を設定します。これらの拡張機能は、証明書要求に指定するか、プロファイルのデフォルト定義から取得した後、制約によって適用できます。

証明書拡張機能がプロファイルで追加または識別されるには、拡張機能に対応する **デフォルト** を追加し、証明書拡張機能が要求で設定されていない場合にデフォルト値を設定します。たとえば、Basic Constraints Extension は、証明書が CA 署名証明書であるかどうか、CA の下で設定できる従属 CA の最大数、および拡張機能が重要 (必須) であるかどうかを識別します。

```
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
```

拡張機能は、**constraints** と呼ばれる証明書要求に必要な値を設定することもできます。リクエストの内容がセット制約と一致しない場合、リクエストは拒否されます。制約は通常、拡張機能のデフォルトに対応しますが、常にそうとは限りません。以下に例を示します。

```
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
```

-

注記

ユーザーが指定する拡張機能を証明書要求に組み込むのを許可し、プロファイルでシステム定義のデフォルトを無視するには、プロファイルに、「User Supplied Extension Default」 で説明されているユーザー Supplied Extension Default を含める必要があります。

3.1.3. 入力および出力

証明書を受信するために送信する必要がある入力セット情報。Common Criteria 環境では、有効なすべてのプロファイルの ***input.i1.class_id*** パラメーターを **cmcCertReqInputImpl** に設定します。

```
input.i1.class_id=cmcCertReqInputImpl
```

プロファイルで設定された出力では、発行された証明書の形式を定義します。Common Criteria 環境では、有効なすべてのプロファイルの ***output.o1.class_id*** パラメーターを **certOutputImpl** に設定します。

```
output.o1.class_id=CertOutputImpl
```

Common Criteria 準拠の Certificate System 環境では、ユーザーはエンドエンティティーインターフェイスからアクセスされる **/ca/ee/ca/profileSubmitUserSignedCMCFull** サーブレットを介してプロファイルにアクセスします。

3.2. 証明書プロファイルの設定

証明書システムでは、登録プロファイルを追加、削除、および変更できます。

- PKI コマンドラインインターフェイスの使用
- Java ベースの管理コンソールの使用

本セクションでは、各メソッドに関する情報を提供します。

3.2.1. PKI コマンドラインインターフェイスを使用した証明書の登録プロファイルの管理

本セクションでは、**pki** ユーティリティーを使用して証明書プロファイルを管理する方法を説明します。詳細は、pki-ca-profile(1) の man ページを参照してください。

注記

RAW 形式の使用が推奨されます。プロファイルの各属性およびフィールドの詳細は、Red Hat Certificate System 計画、インストール、およびデプロイメントガイドの証明書プロファイルの作成および編集を参照してください。

3.2.1.1. 証明書プロファイルの有効化および無効化

証明書プロファイルを編集する前に、無効にする必要があります。変更が完了したら、プロファイルを再度有効にできます。

注記

CA エージェントのみが、証明書プロファイルを有効化および無効化できます。

たとえば、**caCMCECserverCert** 証明書プロファイルを無効にするには、次のコマンドを実行します。

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

たとえば、**caCMCECserverCert** 証明書プロファイルを有効にするには、次のコマンドを実行します。

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

3.2.1.2. Raw 形式の証明書プロファイルの作成

新規プロファイルを raw 形式で作成するには、次のコマンドを実行します。

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

注記

raw 形式で、以下のように新しいプロファイル ID を指定します。

```
profileId=profile_name
```

3.2.1.3. RAW 形式での証明書プロファイルの編集

CA 管理者は、設定ファイルを手動でダウンロードせずに、RAW 形式で証明書プロファイルを編集できます。

たとえば、**caCMCECserverCert** プロファイルを編集するには、次のコマンドを実行します。

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

このコマンドは、プロファイル設定を RAW 形式で自動的にダウンロードし、**VI** エディターで開きます。エディターを閉じると、サーバーでプロファイル設定が更新されます。

プロファイルの編集後に CA を再起動する必要はありません。

重要

プロファイルを編集する前に、プロファイルを無効にします。詳細は、「証明書プロファイルの有効化および無効化」 を参照してください。

例3.2 RAW 形式での証明書プロファイルの編集

たとえば、**caCMCserverCert** プロファイルを編集して、ユーザーが提供する複数の拡張機能を許可するには、次を行います。

1. CA エージェントであるプロファイルを無効にします。

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

2. プロファイルを CA 管理者として編集します。

   a. **VI** エディターでプロファイルをダウンロードして開きます。

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

   b. 設定を更新して、拡張機能を受け入れます。詳細は、???を参照してください。

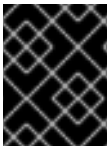
3. プロファイルを CA エージェントとして有効にします。

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

3.2.1.4. 証明書プロファイルの削除

証明書プロファイルを削除するには、次のコマンドを実行します。

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



重要

プロファイルを削除する前に、プロファイルを無効にします。詳細は、「証明書プロファイルの有効化および無効化」 を参照してください。

3.2.2. Java ベースの管理コンソールを使用した証明書の登録プロファイルの管理

3.2.2.1. CA コンソールを使用した証明書プロファイルの作成

セキュリティー上の理由から、Certificate System は、ロールの分離を強制します。これにより、既存の証明書プロファイルは、エージェントによって許可された後にのみ管理者が編集できます。新しい証明書プロファイルを追加するか、既存の証明書プロファイルを変更するには、管理者として以下の手順を実施します。

1. Certificate System CA サブシステムコンソールにログインします。

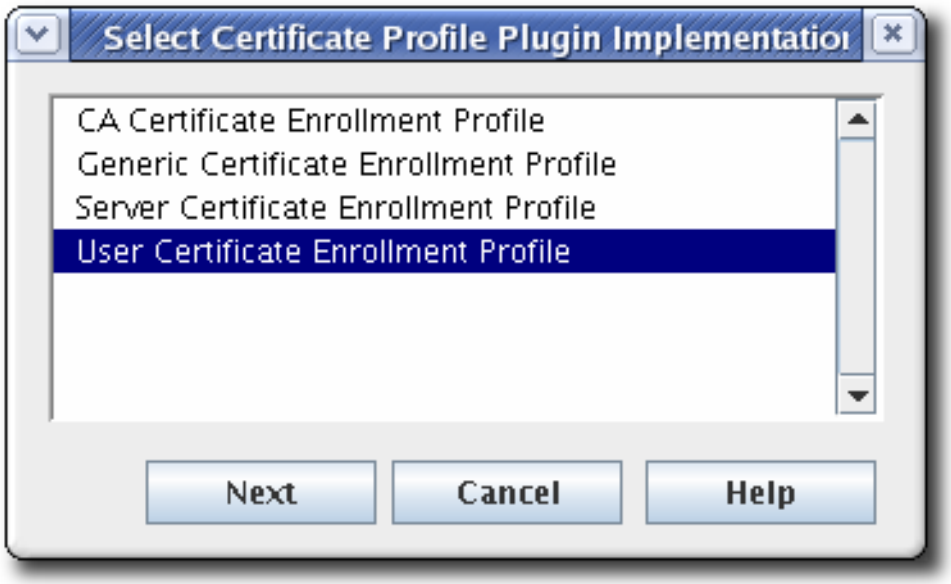
   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで **Certificate Manager** を選択し、**Certificate Profiles** を選択します。

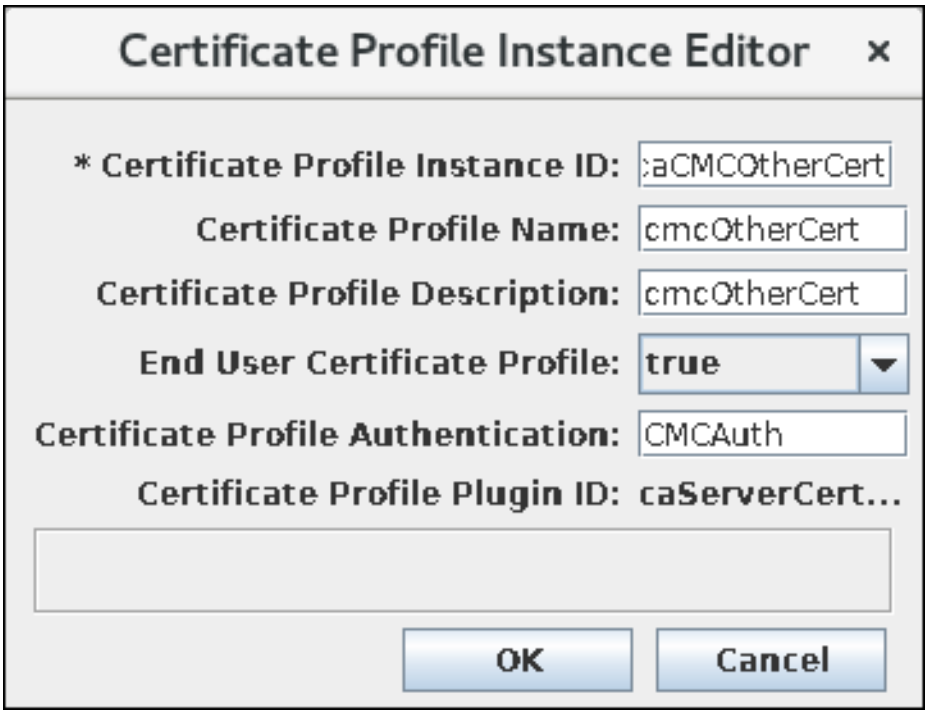
   設定した証明書プロファイルを一覧表示する **Certificate Profile Instances Management** タブが開きます。

3. 新しい証明書プロファイルを作成するには、**Add** をクリックします。

   Select Certificate Profile Plugin Implementation ウィンドウで、プロファイルが作成される証明書のタイプを選択します。



4. **Certificate Profile Instance Editor** にプロファイル情報を入力します。



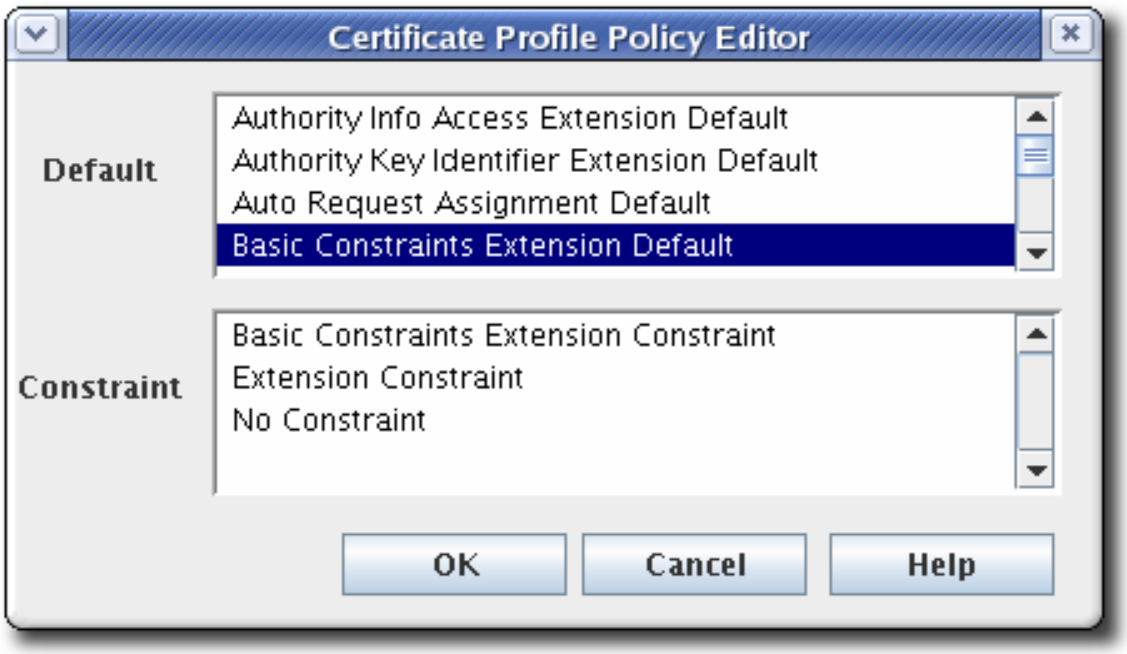
- **Certificate Profile Instance ID**。この ID は、システムがプロファイルの特定に使用する ID です。
- **証明書プロファイル名**これは、ユーザーが分かりやすいプロファイルの名前です。
- **Certificate Profile Description**。
- **End User Certificate Profile**。これにより、リクエストがプロファイルの入力フォームを介して行われる必要があるかどうかが設定されます。これは通常 **true** に設定されます。これを **false** に設定すると、証明書プロファイルの入力ページではなく、Certificate Manager の証明書プロファイルフレームワークを介して署名済みリクエストが処理できるようになります。
- **証明書プロファイル認証**これにより、認証方法が設定されます。認証インスタンスのインスタンス ID を指定して、自動認証が設定されます。このフィールドが空の場合、認証方法はエージェントで承認される登録になります。要求はエージェントサービスインターフェ

イスの要求キューに送信されます。

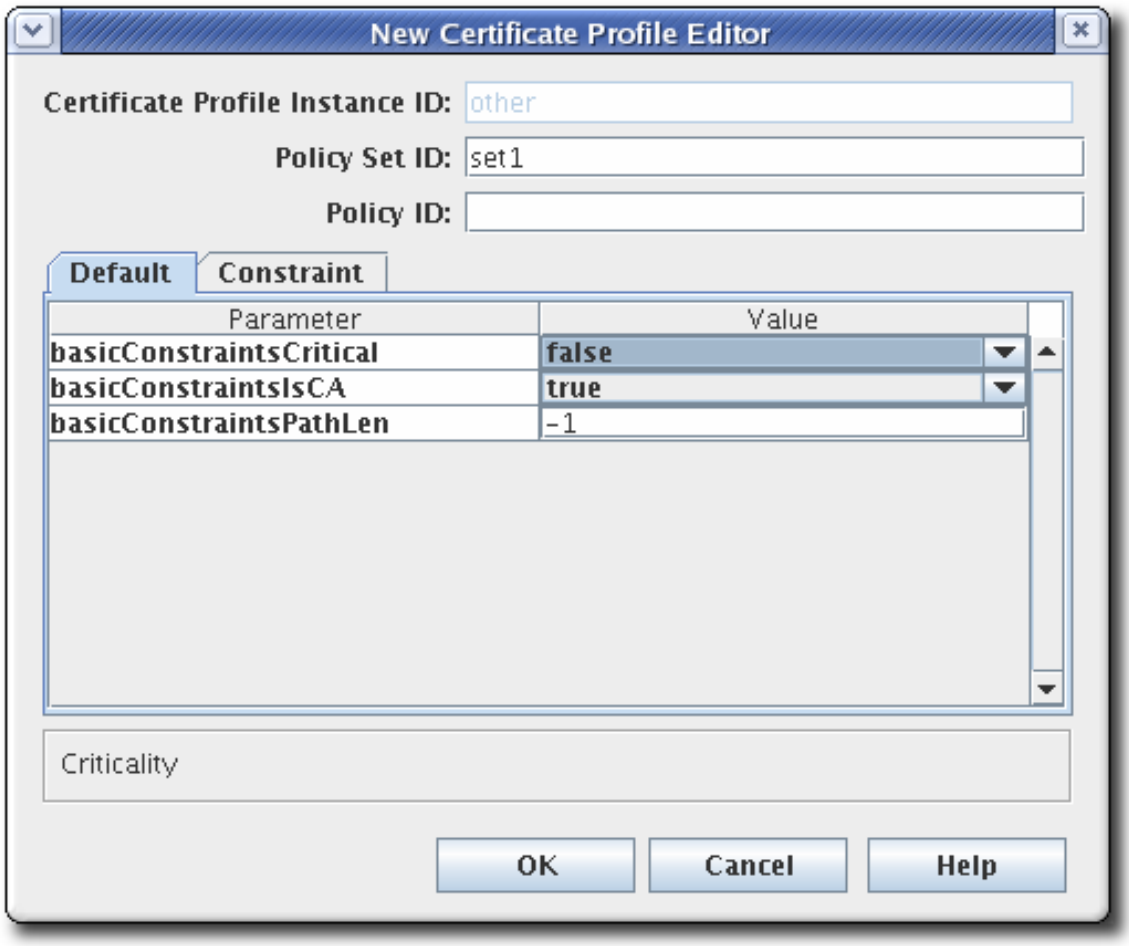
TMS サブシステム用でない限り、管理者は次の認証プラグインのいずれかを選択する必要があります。

- **CMCAuth**: CA エージェントが登録要求を承認し、送信する必要がある場合に、このプラグインを使用します。
- **CMCUserSignedAuth**: このプラグインを使用して、エージェント以外のユーザーが独自の証明書を登録できるようにします。

5. **OK** をクリックします。プラグインエディターが閉じられ、新しいプロファイルが profiles タブに一覧表示されます。
6. 新規プロファイルのポリシー、入力、および出力を設定します。一覧から新しいプロファイルを選択し、**Edit/View** をクリックします。
7. **Certificate Profile Rule Editor** ウィンドウの **Policies** タブでポリシーを設定します。**ポリシー** タブには、プロファイルタイプに対してデフォルトで設定されているポリシーが一覧表示されます。
   a. ポリシーを追加するには、**Add** をクリックします。

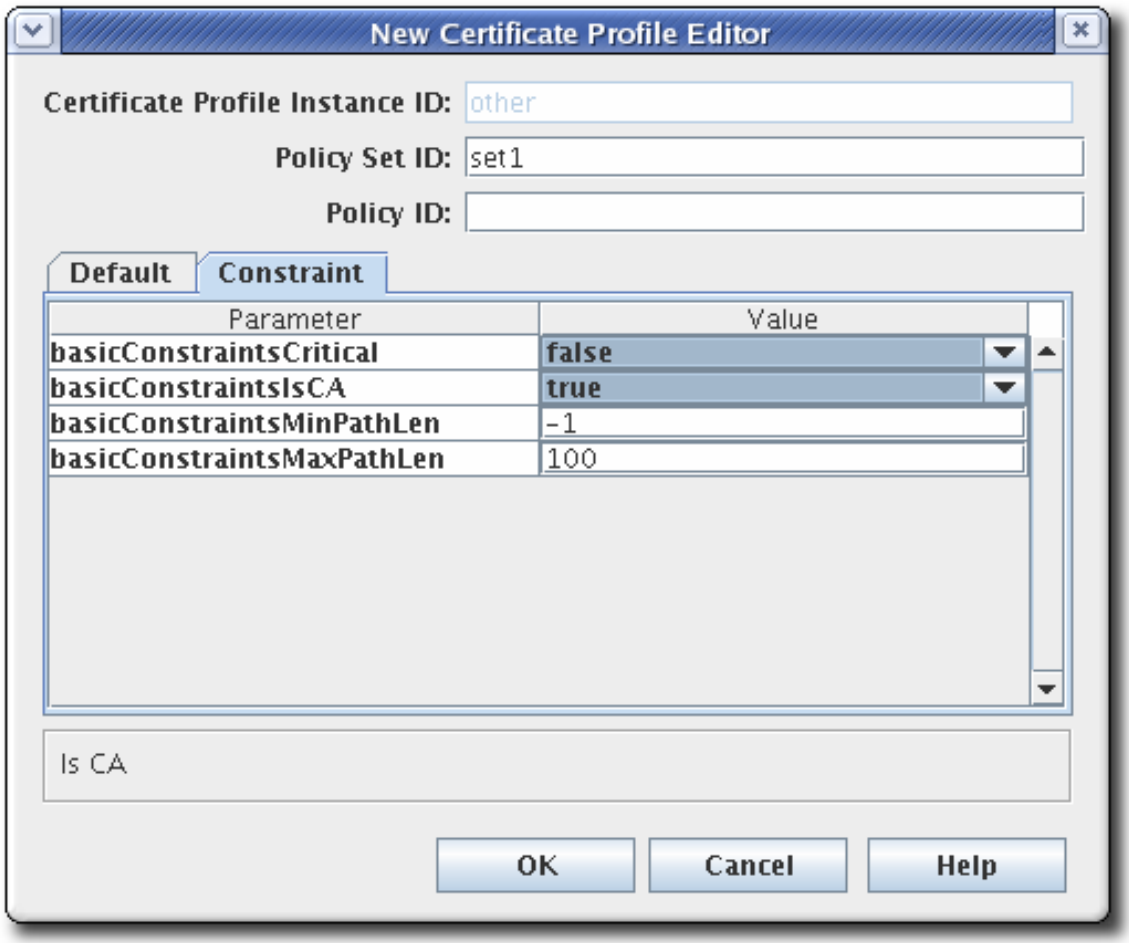


   b. **Default** フィールドからデフォルトを選択して、**Constraints** フィールドでそのポリシーに関連付けられた制約を選択し、**OK** をクリックします。



c. ポリシー設定 ID を入力します。デュアルキーペアを発行する場合には、個別のポリシーセットで、各証明書に関連付けられたポリシーを定義します。次に、証明書プロファイルポリシー ID と、証明書プロファイルポリシーの名前または識別子を入力します。

d. **Defaults** タブおよび **Constraints** タブでパラメーターを設定します。



Defaults は、証明書要求に設定する属性を定義します。これにより、証明書の内容が決定されます。これらは、拡張、有効期間、または証明書に含まれるその他のフィールドのいずれかになります。**制約** は、デフォルト値の有効な値を定義します。

各デフォルトまたは制約の詳細は、「デフォルトの参照」 および 「制約の参照」 を参照してください。

既存のポリシーを変更するには、ポリシーを選択し、**Edit** をクリックします。次に、そのポリシーのデフォルトおよび制約を編集します。

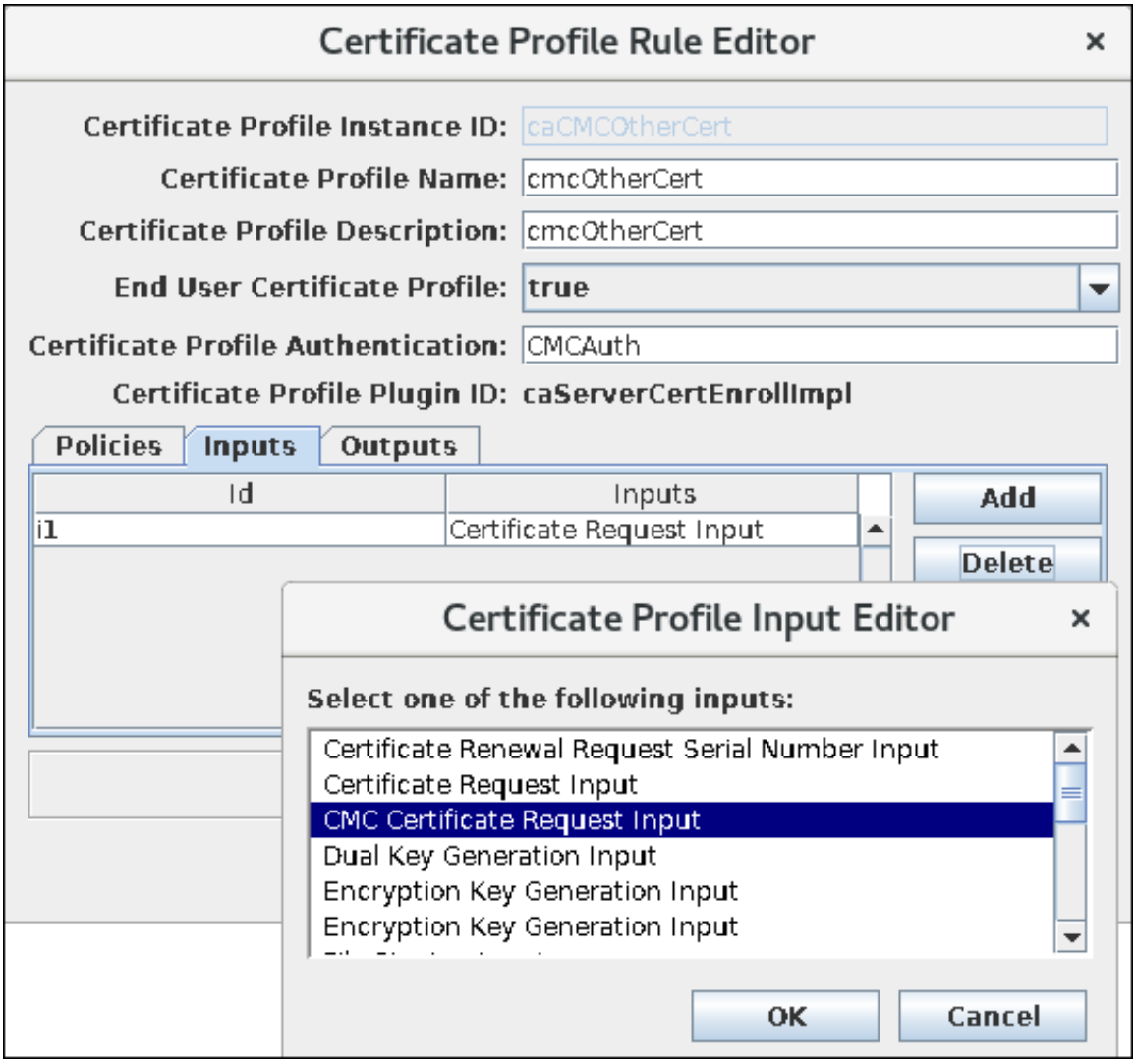
ポリシーを削除するには、ポリシーを選択し、**Delete** をクリックします。

8. **Certificate Profile Rule Editor** ウィンドウの **Inputs** タブに入力を設定します。プロファイルには複数の入力タイプが存在します。

注記

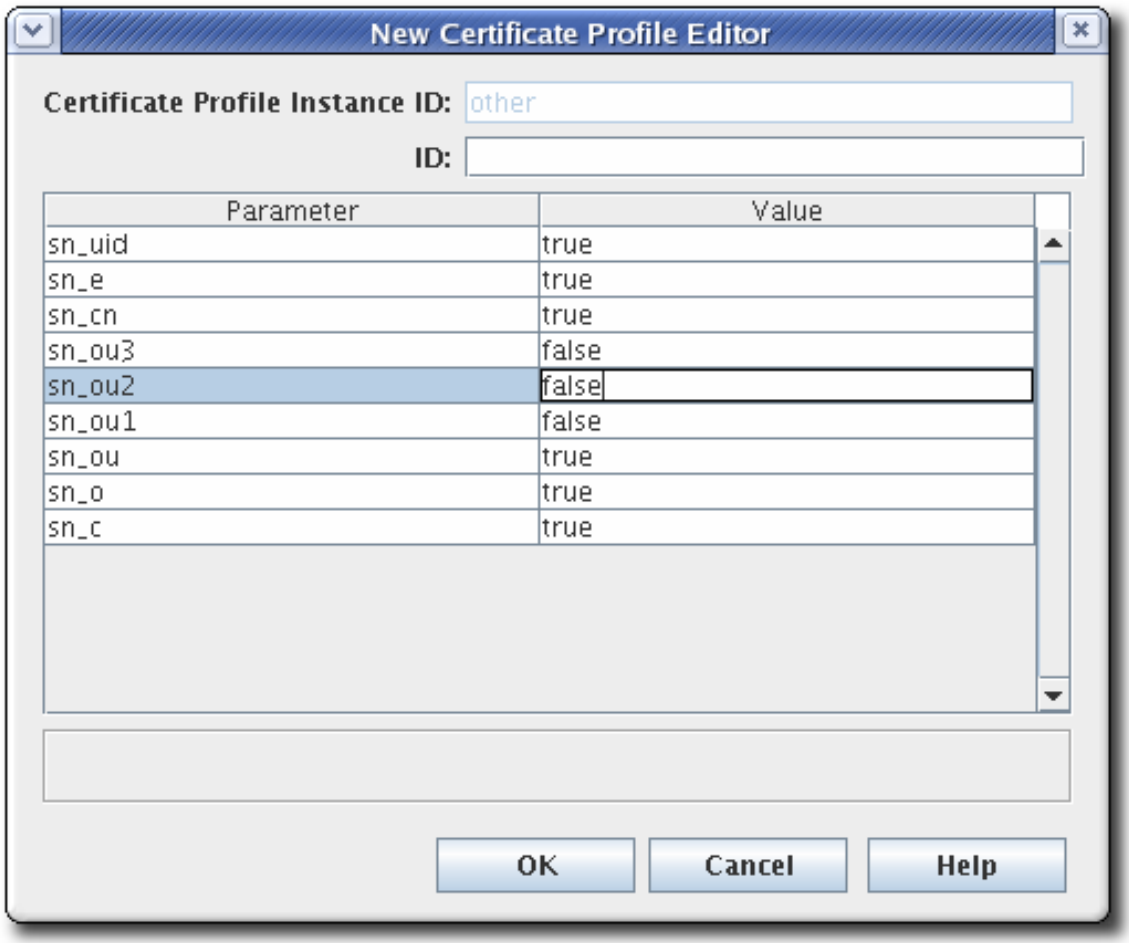
TMS サブシステムにプロファイルを設定しない場合は、**cmcCertReqInput** のみを選択して **Delete** ボタンをクリックし、他のプロファイルを削除します。

a. 入力を追加するには、**Add** をクリックします。



b. 一覧から入力を選択し、**OK** をクリックします。デフォルト入力の完全な詳細については、「入力の参照」 を参照してください。

c. **New Certificate Profile Editor** ウィンドウが開きます。入力 ID を設定して、**OK** をクリックします。



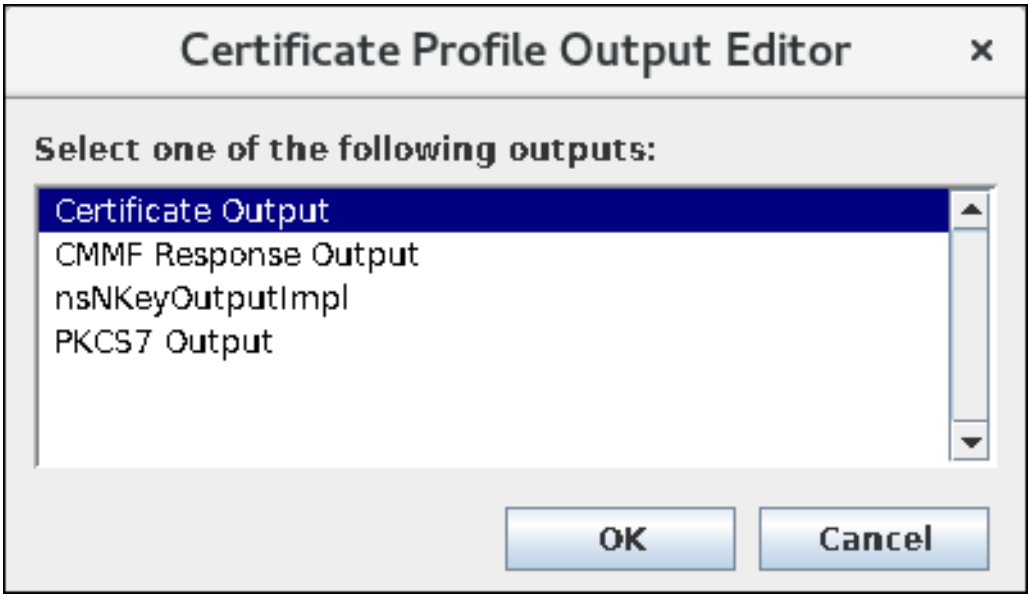
入力は追加および削除が可能です。入力の編集を選択することは可能ですが、入力にはパラメーターやその他の設定がないため、設定するものはありません。

入力を削除するには、入力を選択して **Delete** をクリックします。

9. **Certificate Profile Rule Editor** ウィンドウの **Outputs** タブで、出力を設定します。

   自動認証方式を使用する証明書プロファイルには、出力を設定する必要があります。エージェントが承認した認証を使用する証明書プロファイルには、出力を設定する必要はありません。Certificate Output タイプはすべてのプロファイルでデフォルトで設定され、カスタムプロファイルに自動的に追加されます。

   TMS サブシステムにプロファイルを設定しない限り、**certOutput** のみを選択します。



出力を追加または削除できます。出力の編集を選択することは可能ですが、出力にはパラメーターやその他の設定がないため、設定するものはありません。

a. 出力を追加するには、**Add** をクリックします。

b. 一覧から出力を選択して **OK** をクリックします。

c. 出力の名前または識別子を指定して、**OK** をクリックします。

この出力は出力タブに一覧表示されます。これを編集して、この出力のパラメーターに値を指定できます。

出力を削除するには、一覧から出力を選択して **Delete** をクリックします。

10. CA を再起動して、新規プロファイルを適用します。

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

11. プロファイルを管理者として作成したら、CA エージェントはエージェントサービスページでプロファイルを承認してプロファイルを有効にする必要があります。

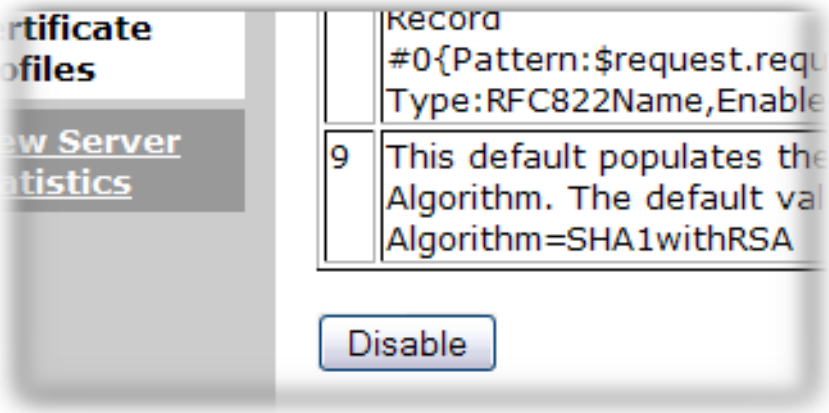
a. CA のサービスページを開きます。

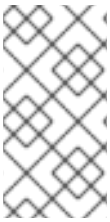
```
https://server.example.com:8443/ca/services
```

b. **証明書プロファイルの管理** リンクをクリックします。このページには、アクティブと非アクティブの両方で、管理者によって設定されたすべての証明書プロファイルが一覧表示されます。

c. 承認する証明書プロファイルの名前をクリックします。

d. ページの下部で、**Enable** ボタンをクリックします。





注記

このプロファイルを TPS で使用する場合は、プロファイルタイプを認識するように TPS を設定する必要があります。これは 11.1.4. にあります。Red Hat Certificate System の計画、インストール、およびデプロイメントガイドのスマートカード CA プロファイルの管理

プロファイルの承認方法は、Red Hat Certificate System の計画、インストール、およびデプロイメントガイドのファイルシステムの証明書プロファイルの作成および編集のセクションで説明されているように、コマンドラインでのみプロファイルに追加できます。

3.2.2.2. コンソールでの証明書プロファイルの編集

たとえば、既存の証明書プロファイルを修正するには、以下の手順に従います。

1. エージェントサービスページにログインし、プロファイルを無効にします。

   エージェントで証明書プロファイルを有効にすると、その証明書プロファイルは **Certificate Profile Instance Management** タブで有効とマークされ、コンソールを介して証明書プロファイルはいつでも編集できません。

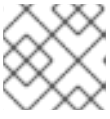
2. Certificate System CA サブシステムコンソールにログインします。

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

3. **Configuration** タブで **Certificate Manager** を選択し、**Certificate Profiles** を選択します。

4. 証明書のプロファイルを選択して、**Edit/View** をクリックします。

5. **Certificate Profile Rule Editor** ウィンドウが表示されます。多くのデフォルト、制約、入力、または出力が変更されています。

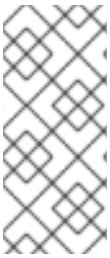
   注記

   プロファイルインスタンス ID は変更しないでください。

   必要に応じて、ウィンドウの隅の 1 つを引っ張って、ウィンドウを拡大します。

6. CA を再起動して変更を適用します。

7. エージェントサービスページで、プロファイルを再度有効にします。



注記

エージェントによって承認されない証明書プロファイルを削除します。**Certificate Profile Instance Management** タブに表示される証明書プロファイルも、エージェントサービスインターフェイスに表示されます。プロファイルがすでに有効になっている場合は、プロファイルリストから削除する前に、エージェントによって無効にする必要があります。

3.2.3. 証明書の登録プロファイルの一覧表示

以下の事前定義済みの証明書プロファイルを使用し、Certificate System CA のインストール時にこの環境で使用できます。これらの証明書プロファイルは、最も一般的なタイプの証明書用に設計されており、一般的なデフォルト、制約、認証方法、入力、および出力を提供します。

サポートされるプロファイルの一覧は、「CMC 認証プラグイン」 を参照してください。

コマンドラインで利用可能なプロファイルを一覧表示するには、**pki** ユーティリティーを使用します。以下に例を示します。

```
# 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
```

詳細は、pki-ca-profile(1) の man ページを参照してください。

3.2.4. 証明書登録プロファイルの詳細表示

たとえば、**caECFullCMCUserSignedCert** などの特定の証明書プロファイルを表示するには、次のコマンドを実行します。

```
$ 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
```

たとえば、**caECFullCMCUserSignedCert** などの特定の証明書プロファイルを表示するには、raw 形式で次のコマンドを実行します。

```
$ 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

...
```

詳細は、pki-ca-profile(1) の man ページを参照してください。

3.3. プロファイルでの鍵のデフォルトの定義

証明書プロファイルの作成時に **サブジェクトキー識別子のデフォルトの前にキーのデフォルトを追加する必要があります**。Certificate System は、サブジェクトキー識別子のデフォルトを作成または適用する前にキーのデフォルトで鍵制約を処理するため、鍵がまだ処理されていない場合は、サブジェクト名に鍵の設定に失敗します。

たとえば、object-signing プロファイルでは両方のデフォルト値を定義できます。

```
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
```

policyset リストでは、サブジェクトキー識別子のデフォルト (**p3**) の前にキーのデフォルト (**p11**) を指定する必要があります。

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

3.4. 更新を有効にするためのプロファイルの設定

証明書を **更新** すると、元の証明書と同じ公開鍵を使用して、証明書が再生成されます。証明書の更新は、単に新しい鍵を生成して新しい証明書をインストールするよりも望ましい場合があります。例えば、新しい CA 署名証明書を作成する場合、その CA が発行し署名したすべての証明書を再発行しなければなりません。CA 署名証明書が更新された場合、発行されたすべての証明書は有効である。更新された証明書は元の証明書と同じで、更新された有効期間と有効期限のみになります。

ここでは、更新のためのプロファイルの設定方法について説明します。

3.4.1. 更新について

更新された証明書は元の証明書と同一であるため、証明書の更新は、多くの種類の証明書、特に CA 署名証明書の有効期限を処理する上で、よりシンプルでクリーンなオプションとなります。

3.4.1.1. 更新の流れ

証明書を更新する方法は 2 つあります。証明書の **再生成** は、証明書の元の鍵、プロファイル、要求を取り出し、同一の鍵を用いて、新しい有効期間と有効期限を持つ新しい証明書を再作成する。証明書の **キーの再生成**は、元のプロファイルから同じ情報で証明書要求を提出し，新しいキーペアを生成します。

更新を許可するプロファイルは、多くの場合、update **GracePeriodConstraint** エントリーに含まれます。以下に例を示します。

```
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
```

3.4.1.1.1. 同じ鍵を使用した更新

更新に同じキーの送信を可能にするプロファイルで、**uniqueKeyConstraint** エントリーの ***allowSameKeyRenewal*** パラメーターが **true** に設定されています。以下に例を示します。

```
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
```

3.4.1.1.2. 新しい鍵を使用した更新

新しい鍵で証明書を更新するには、新しいキーで同じプロファイルを使用します。Certificate System は、新しい証明書の要求の署名に使用されるユーザー署名証明書の **subjectDN** を使用します。

3.5. 証明書の署名アルゴリズムの設定

CA の署名証明書は、CA でサポートされる公開鍵アルゴリズムに問題がある証明書に署名できます。たとえば、ECC 署名証明書は、ECC アルゴリズムと RSA アルゴリズムの両方が CA でサポートされている限り、ECC 証明書要求と RSA 証明書要求の両方に署名できます。RSA 署名証明書は EC キーを使用して PKCS # 10 要求に署名できますが、CA が CRMF 所有証明 (POP) を検証するために ECC モジュールを使用できない場合は、EC キーを使用して CRMF 証明書要求に署名できない場合があります。

ECC および RSA は、公開鍵の暗号化と署名アルゴリズムです。両方の公開鍵アルゴリズムは、異なる暗号スイートをサポートします。これは、データの暗号化と復号に使用されるアルゴリズムです。CA 署名証明書の機能には、対応している暗号スイートのいずれかを使用して、証明書を発行し、署名することです。

各プロファイルは、CA がそのプロファイルで処理される証明書の署名に使用する暗号化スイートを定義できます。署名アルゴリズムが設定されていない場合、プロファイルはデフォルトの署名アルゴリズムを使用します。

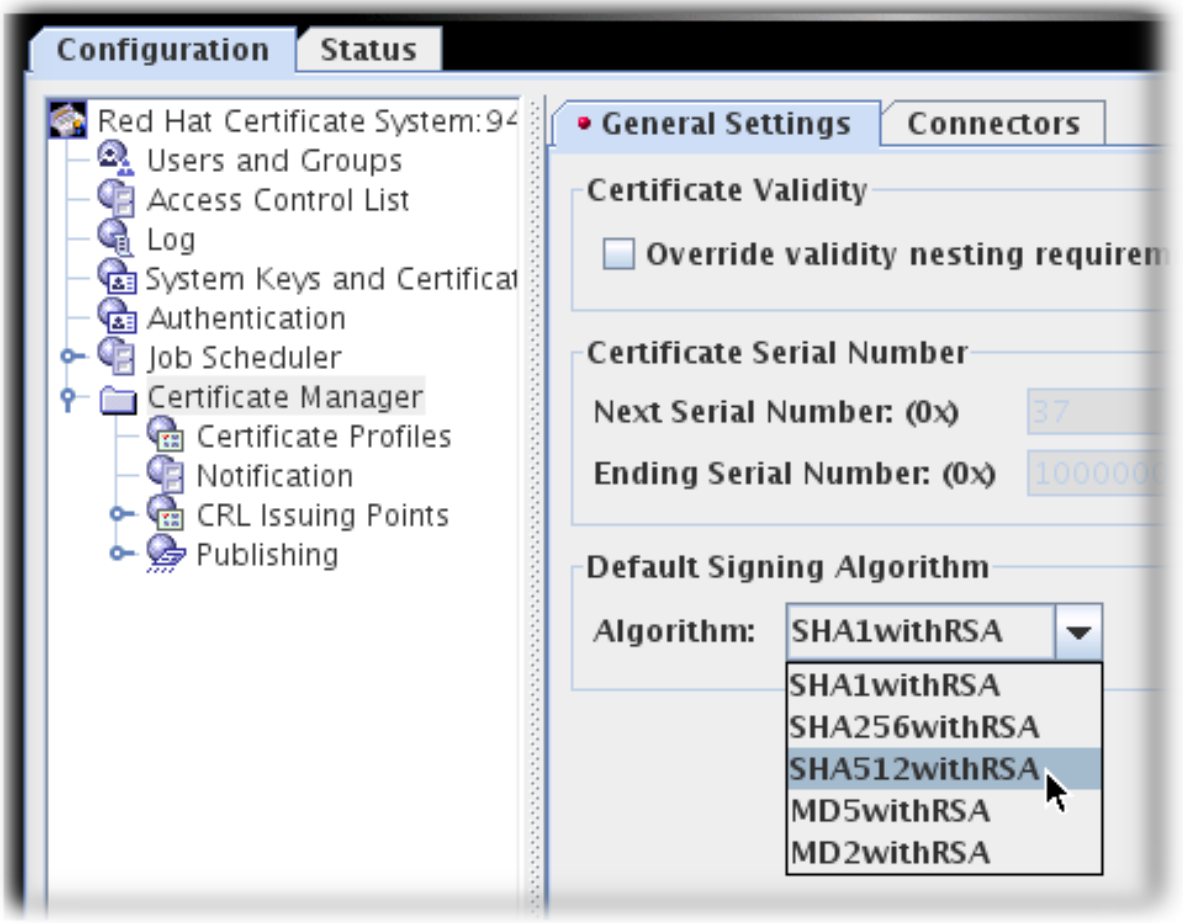
3.5.1. CA のデフォルト署名アルゴリズムの設定

1. CA コンソールを開きます。

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

2. **Configuration** タブで、**Certificate Manager** ツリーを展開します。

3. **General Settings** タブで、**Algorithm** ドロップダウンメニューで使用するアルゴリズムを設定します。



3.5.2. プロファイルでの署名アルゴリズムのデフォルトの設定

各プロファイルには、Signing Algorithm Default 拡張が定義されています。デフォルトには、デフォルトのアルゴリズムと、証明書要求で別のアルゴリズムが指定されている場合に許可されるアルゴリズムのリストの 2 つの設定があります。署名アルゴリズムが指定されていない場合、プロファイルは CA のデフォルトとして設定されているものを使用します。

プロファイルの **.cfg** ファイルで、アルゴリズムは 2 つのパラメーターで設定されます。

```
policyset.cmcUserCertSet.8.constraint.class_id=signingAlgConstraintImpl
policyset.cmcUserCertSet.8.constraint.name=No Constraint
policyset.cmcUserCertSet.8.constraint.params.signingAlgsAllowed=SHA256withRSA,SHA512withRSA,S
HA256withEC,SHA384withRSA,SHA384withEC,SHA512withEC
policyset.cmcUserCertSet.8.default.class_id=signingAlgDefaultImpl
policyset.cmcUserCertSet.8.default.name=Signing Alg
policyset.cmcUserCertSet.8.default.params.signingAlg=-
```

コンソールから Signing Algorithm Default を設定するには、以下を行います。

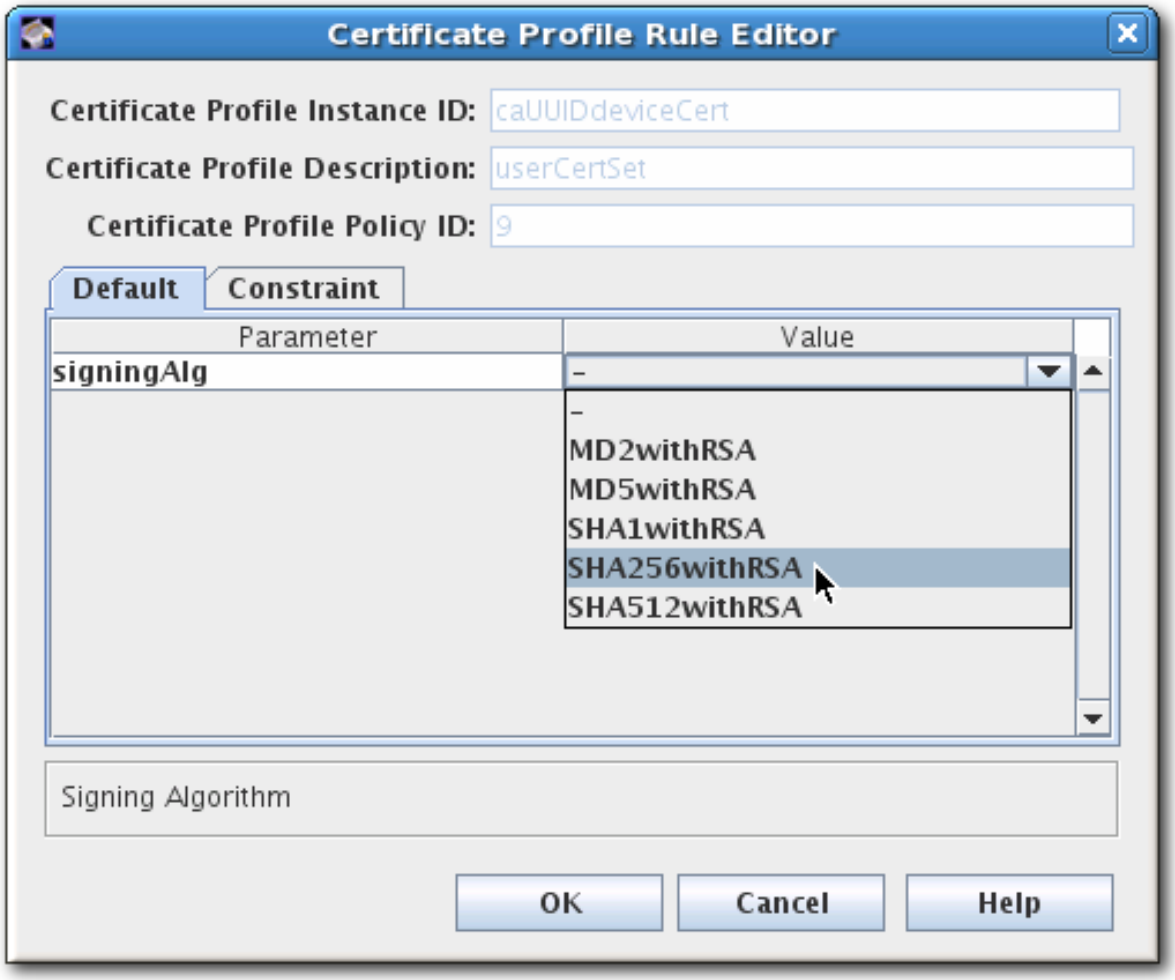
注記

プロファイルを編集する前に、エージェントにより最初に無効にする必要があります。

1. CA コンソールを開きます。

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

2. **Configuration** タブで、**Certificate Manager** ツリーを展開します。
3. **Certificate Profiles** 項目をクリックします。
4. **Policies** タブをクリックします。
5. **Signing Alg ポリシー** を選択して **Edit** ボタンをクリックします。
6. デフォルトの署名アルゴリズムを設定するには、**Defaults** タブで値を設定します。これが **-** に設定されている場合、プロファイルは CA のデフォルトを使用します。



7. 証明書要求で許可される署名アルゴリズムの一覧を設定するには、**Constraints** タブを開き、**singiningAlgsAllowed** の **Value** フィールドでアルゴリズムの一覧を設定します。

   制約に使用できる値は、「アルゴリズム制約の署名」 に記載されています。

3.6. CA 関連プロファイルの管理

証明書プロファイルと拡張機能を使用して、下位 CA が証明書を発行する方法にルールを設定する必要があります。これには 2 つの部分があります。

- CA 署名証明書の管理
- 発行ルールの定義

3.6.1. CA 証明書での制限の設定

下位 CA が作成されると、ルート CA は下位 CA に制限または制限を課できます。たとえば、ルート CA は、CA 署名証明書の Basic Constraints 拡張機能の pathLenConstraint フィールドを設定することにより、有効な認証パスの最大深度 (新しい CA の下にチェーンできる下位 CA の数) を指定できます。

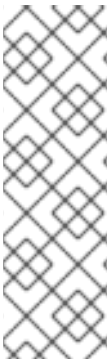
証明書チェーンは、通常エンティティー証明書、ゼロまたは中間 CA 証明書、ルート CA 証明書で設定されます。ルート CA 証明書は、自己署名型または外部の信頼できる CA によって署名されます。ルート CA 証明書は、信頼できる CA として証明書データベースに読み込まれます。

証明書の交換は、TLS ハンドシェイクの実行時、S/MIME メッセージの送信時、または署名済みオブジェクトを送信するときに行われます。ハンドシェイクの一部として、送信者は、サブジェクト証明書と、サブジェクト証明書を信頼されたルートにリンクするために必要な中間 CA 証明書を送信する必要があります。証明書チェーンが適切に証明書を有効にするには、以下のプロパティーが必要です。

- CA 証明書には、基本的な制約の拡張子が必要です。
- CA 証明書の鍵用途拡張に keyCertSign ビットが設定されている必要があります。
- CA が新しい鍵を生成する場合は、すべてのサブジェクト証明書に認証局キー識別子の拡張子を追加する必要があります。この拡張機能は、これらの証明書を古い CA 証明書と区別するのに役立ちます。CA 証明書には Subject Key Identifier 拡張が含まれている必要があります。

証明書とその拡張の詳細は、RFC 5280 で利用可能な 『Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile (RFC 5280)』 を参照してください。

これらの拡張機能は、証明書プロファイルの登録ページで設定できます。デフォルトでは、CA には必須かつ合理的な設定設定が含まれますが、これらの設定をカスタマイズすることは可能です。



注記

この手順では、CA が使用する CA 証明書プロファイルを編集して、下位 CA に CA 証明書を発行する方法を説明します。

CA インスタンスの初期設定時に使用されるプロファイルは、**/var/lib/pki/*instance_name*/ca/conf/caCert.profile** です。このプロファイルは、**pkiconsole** で編集することはできません (インスタンスを設定する前のみ利用可能)。テキストエディターで CA を設定する前に、テンプレートファイルでこのプロファイルのポリシーを編集することができます。

CA が使用する CA 署名証明書プロファイルでデフォルトを変更するには、以下を実行します。

1. プロファイルが有効になっている場合は、編集する前に無効にする必要があります。エージェントサービスページを開き、左側のナビゲーションメニューから **Manage Certificate Profile** を選択してプロファイルを選択し、**Disable profile** をクリックします。
2. CA コンソールを開きます。

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

3. **Configuration** タブの左側のナビゲーションツリーで、**Certificate Manager** を選択し、**Certificate Profiles** を選択します。
4. 右側のウィンドウから caCACert または該当する CA 署名証明書プロファイルを選択して、**Edit/View** をクリックします。
5. **Certificate Profile Rule Editor** の **Policies** タブで、キー使用法または拡張キー使用法拡張機能のデフォルトが存在する場合はそれを選択して編集するか、プロファイルに追加します。

6. 必要に応じて、デフォルトとして、Key Usage または Extended Key Usage Extension Constraint を選択します。

7. CA 証明書のデフォルト値を設定します。詳細は、「Key Usage 拡張機能のデフォルト」および「Extended Key Usage 拡張機能のデフォルト」を参照してください。

8. CA 証明書の制約値を設定します。キー使用拡張に設定する制約はありません。拡張キーの使用の拡張機能の場合は、CA に適切な OID 制約を設定します。詳細は、「Extended Key Usage 拡張機能のデフォルト」を参照してください。

9. プロファイルに変更を加えたら、エージェントサービスページを再度ログインして、証明書プロファイルを再度有効にします。

証明書プロファイルの変更の詳細は、「証明書プロファイルの設定」を参照してください。

3.6.2. 証明書の発行における CA の制限の変更

発行された証明書の制限は、サブシステムの設定後にデフォルトで設定されます。これには、以下が含まれます。

- CA 署名証明書よりも長い有効期間で証明書を発行できるかどうか。デフォルトでは、これを無効にします。
- 証明書の署名に使用される署名アルゴリズム。
- CA が証明書を発行するために使用するシリアル番号の範囲。

下位 CA には、有効期間、証明書の種類、および発行可能な拡張の種類に制約があります。下位 CA はこれらの制約に違反する証明書を発行できますが、これらの制約に違反する証明書を認証するクライアントはその証明書を受け入れません。下位 CA の発行ルールを変更する前に、CA 署名証明書に設定された制約を確認してください。

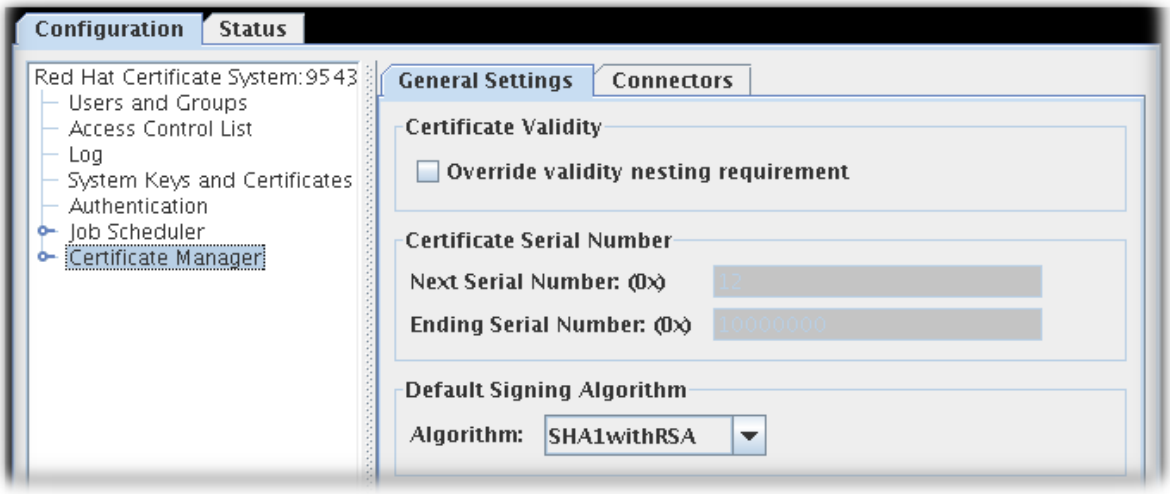
証明書の発行ルールを変更するには、次のコマンドを実行します。

1. 証明書システムコンソールの起動

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

2. **Configuration** タブの左側のナビゲーションツリーで、**Certificate Manager** アイテムを選択します。

図3.1デフォルトでは、非従属 CA の一般設定タブ



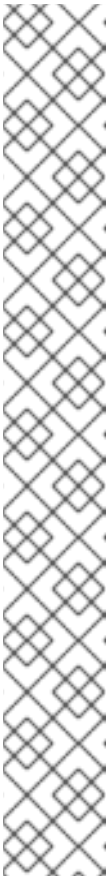
3. デフォルトでは、クローン以外の CA では、**Certificate Manager** メニュー項目の **General Settings** タブに以下のオプションが含まれます。

   - **Override validity nesting requirement**。このチェックボックスでは、Certificate Manager が、CA 署名の証明書有効期間よりも長い有効期間の証明書を発行できるかどうかを設定します。

     このチェックボックスを選択しておらず、CA が CA 署名証明書の有効期間よりも長い期間要求を受け取ると、CA 署名証明書の期限が切れる時点で自動的に終了するように有効期間が切り捨てられます。

   - **Certificate Serial Number**。これらのフィールドは、Certificate Manager が発行する証明書のシリアル番号の範囲を表示します。サーバーは、**Next serial number** を、次に発行する証明書に割り当て、**Ending serial number** を、最後に発行した証明書に割り当てます。

     シリアル番号の範囲により、複数の CA をデプロイでき、各 CA が発行する証明書の数のバランスを取ります。発行者名とシリアル番号の組み合わせは、証明書を一意に識別する必要があります。



注記

クローン CA を使用するシリアル番号の範囲は fluid です。複製されたすべての CA は、次の利用可能な範囲を定義する共通の設定エントリーを共有します。1 つの CA が利用可能な数未満の実行を開始すると、この設定エントリーをチェックし、次の範囲を要求します。エントリーは自動的に更新されます。これにより、次の CA が新規範囲を取得します。

範囲は ***begin*Number*** 属性および ***end*Number*** 属性で定義され、個別の範囲が要求および証明書のシリアル番号に対して定義されます。以下に例を示します。

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

シリアル番号管理は、クローンされていない CA に対して有効にできます。ただし、デフォルトでは、システムが自動的に有効になった場合にシステムのクローンが作成されない限り、シリアル番号の管理が無効になります。

シリアル番号の範囲は、コンソールで手動で更新することはできません。シリアル番号の範囲は読み取り専用フィールドです。

- **署名アルゴリズムのデフォルト。** Certificate Manager が証明書の署名に使用する署名アルゴリズムを指定します。このオプションは、CA の署名鍵タイプが RSA の場合は、**SHA256withRSA** および **SHA512withRSA** です。

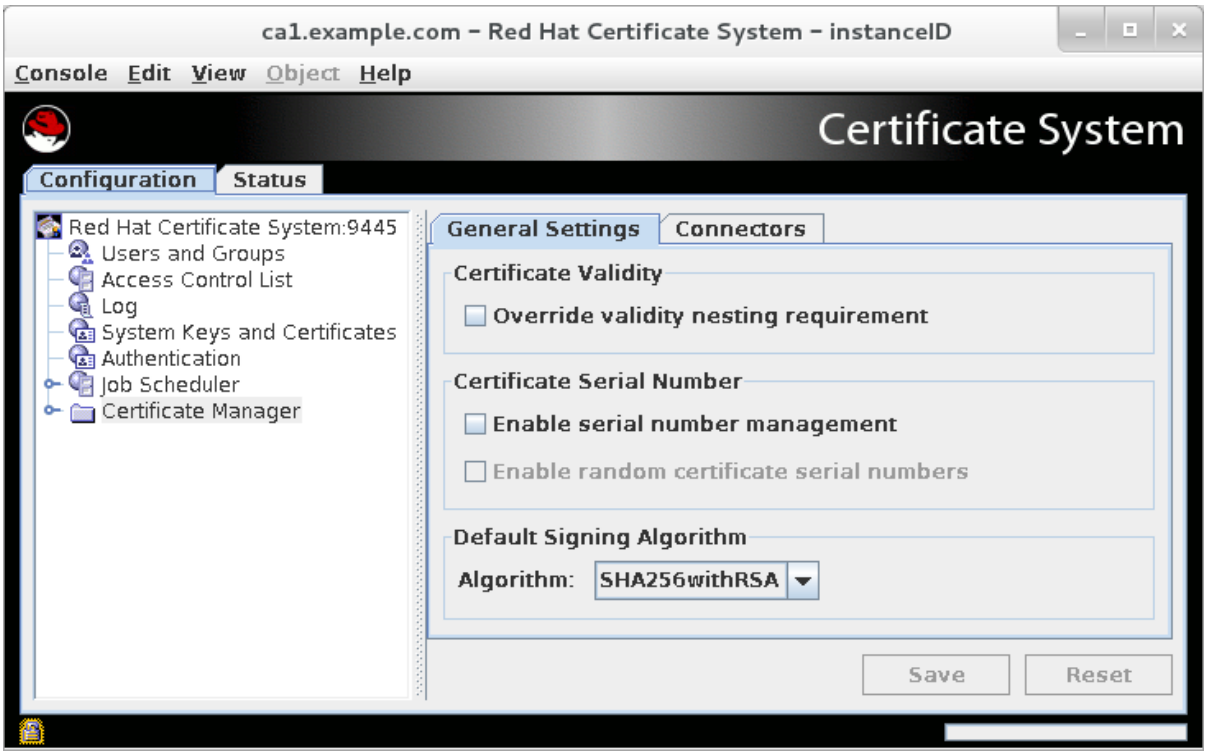
  証明書プロファイル設定に指定された署名アルゴリズムは、ここに設定されたアルゴリズムよりも優先されます。

4. デフォルトでは、クローン作成された CA では、**Certificate Manager** メニュー項目の **General Settings** タブに以下のオプションが含まれます。

   - **ランダムなシリアル番号管理**
   - **Enable random certificate serial numbers**

   両方のチェックボックスを選択します。

図3.2 デフォルトでクローン作成された CA の General Settings タブ



5. **Save** をクリックします。

3.6.3. ランダム証明書のシリアル番号の使用

Red Hat Certificate System には、要求、証明書、レプリカ ID に対するシリアル番号の範囲管理が含まれています。これにより、**Identity Management** (IdM) インストール時のクローン作成を自動化できます。

ハッシュベースの攻撃の可能性が低くなるには、以下の方法を使用できます。

- 攻撃者に予測できない証明書のシリアル番号の一部にする
- ランダムに選択されたコンポーネントを ID に追加する
- それぞれを前後に歪めることにより、攻撃者が有効期限を予測できないようにする

ランダムな証明書のシリアル番号割り当て 方法は、無作為に選択されたコンポーネントを ID に追加します。この方法は以下の通りです。

- クローン作成で機能
- 競合の解決を許可する
- 現在のシリアル番号管理方法との互換性がある
- 管理者、エージェント、およびエンドエンティティーの現在のワークフローと互換性がある
- 連続するシリアル番号管理で既存のバグを修正。

備考

管理者は、証明書のシリアル番号を有効にする必要があります。

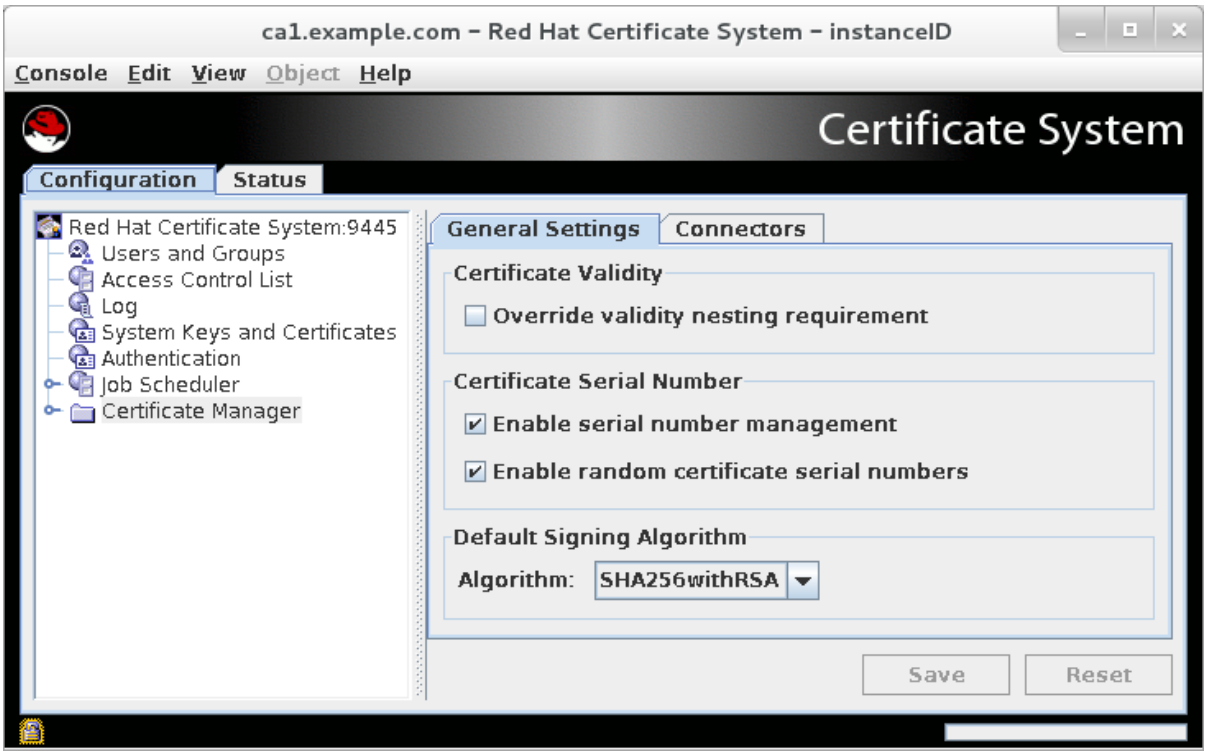
3.6.3.1. ランダム証明書のシリアル番号の有効化

コマンドラインまたはコンソール UI から自動シリアル番号範囲管理を有効にすることができます。

コンソール UI から自動シリアル番号管理を有効にするには、以下を行います。

1. **General Settings** タブで、**Enable serial number management** オプションを選択します。

   図3.3 乱数の割り当てが有効な場合の General Settings タブ



2. **Enable random certificate serial numbers** オプションをオンにします。

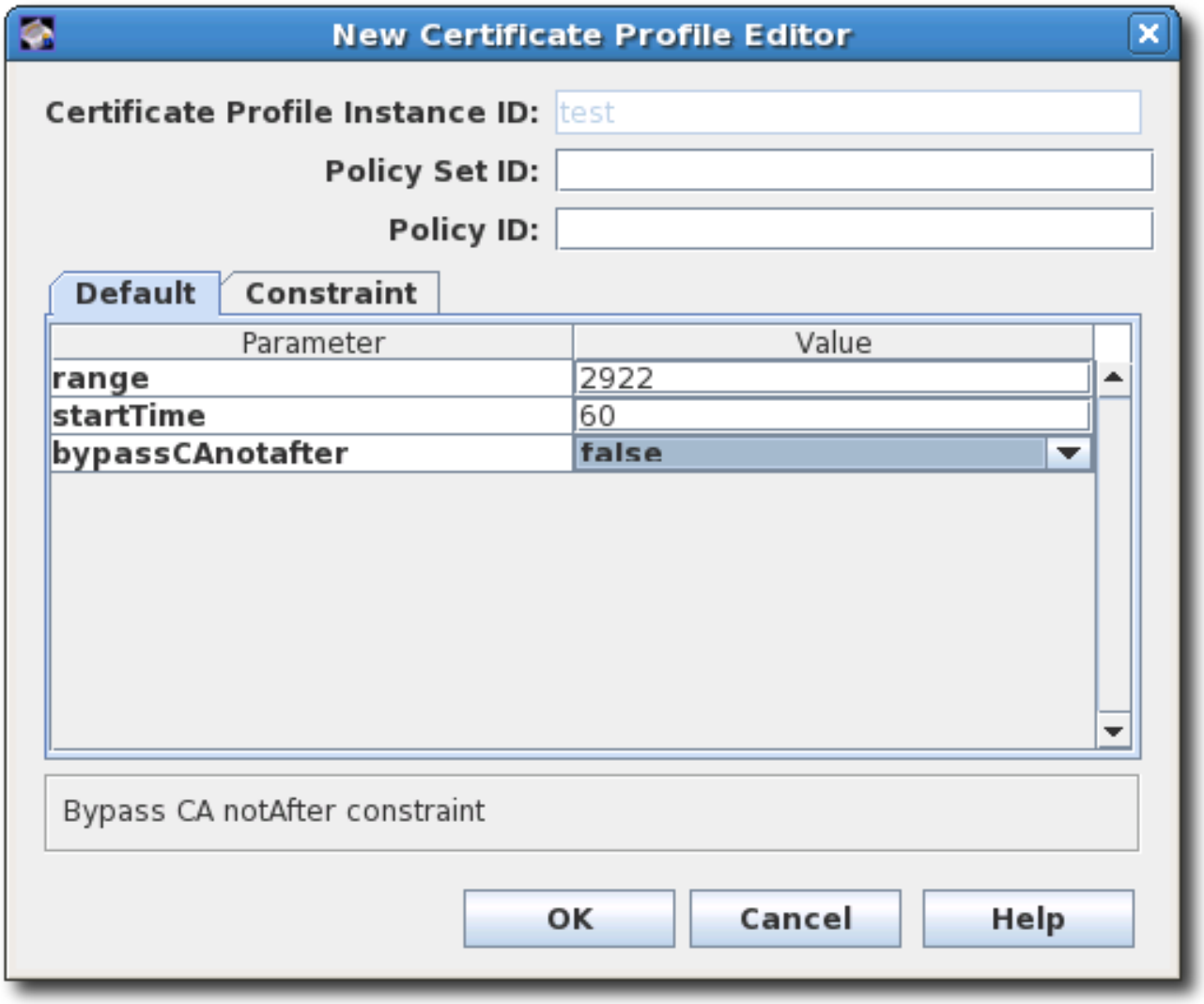
3.6.4. 認証局の有効期間を過ぎた認証局証明書の更新の許可

通常、証明書は、発行先の CA 証明書の有効期限 **後** に終わる有効期間では発行できません。CA 証明書の有効期限が 2015 年 12 月 31 日である場合、証明書はすべて 2015 年 12 月 31 日または 2015 年 12 月 31 日より前に有効期限が切れる必要があります。

このルールは、CA が発行する他の CA 署名証明書に適用されます。これにより、ルート CA 証明書はほとんど更新できなくなります。CA 署名証明書を更新するということは、それ自体の有効期限を過ぎた有効期間が必ず必要になることを意味します。

この動作は CA Validity Default を使用して変更できます。このデフォルト設定では、発行している CA の有効期限 (**notAfter**) の期間を拡張する有効期間で CA 証明書を発行できる設定 (**bypassCAnotafter**) が許可されます。

図3.4 CA 有効性のデフォルト設定



実際のデプロイメントでは、これが意味するのは、ルート CA の CA 証明書は、他の方法では防止できる場合でも更新できるということです。

元の CA の有効期間前に CA 証明書の更新を有効にするには、以下を実行します。

1. **caCACert.cfg** ファイルを開きます。

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

2. CA Validity デフォルトはデフォルトで存在する必要があります。値を **true** に設定して、発行 CA の有効期間を過ぎて CA 証明書を更新できるようにします。

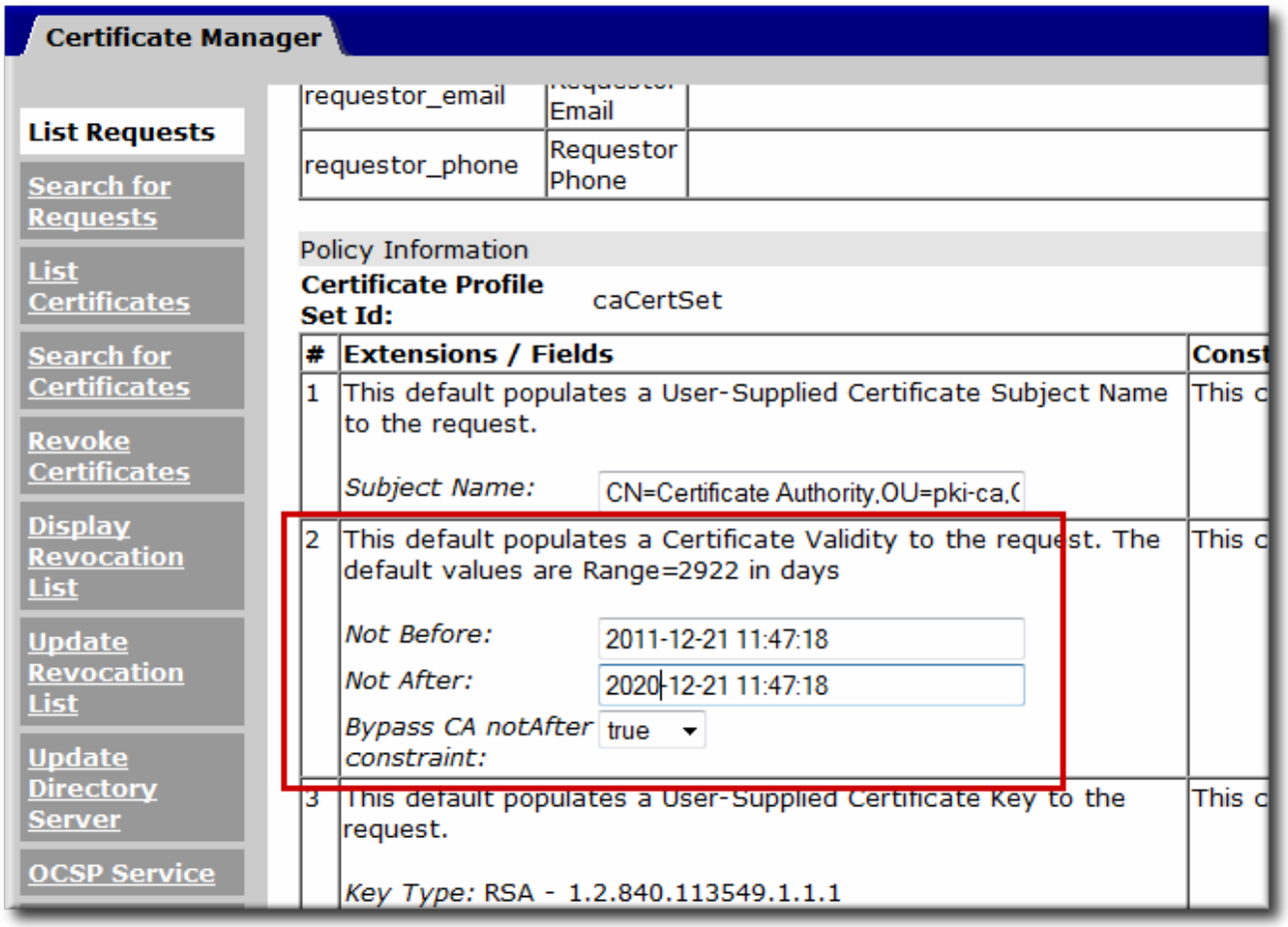
```
policyset.caCertSet.2.default.name=CA Certificate Validity Default
policyset.caCertSet.2.default.params.range=2922
policyset.caCertSet.2.default.params.startTime=0
policyset.caCertSet.2.default.params.bypassCAnotafter=true
```

3. CA を再起動して変更を適用します。

エージェントが更新要求を確認すると、通常の有効期間制約をバイパスすることを可能にする **Extensions/Fields** エリアにオプションがあります。エージェントが **false** を選択すると、プロファイルで **bypassCAnotafter=true** が設定されていても制約が適用されます。**bypassCAnotafter** 値が有効

になっていない時にエージェントが true を選択すると、更新要求は CA によって拒否されます。

図3.5 エージェントサービスページの CA 制約オプションを回避



注記

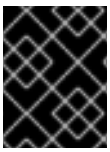
CA Validity のデフォルトは、CA 署名の証明書の更新にのみ適用されます。その他の証明書は、引き続き CA の有効期間内で発行および更新する必要があります。

CA の **ca.enablePastCATime** の個別の設定を使用すると、CA の有効期間を過去に証明書を更新することができます。ただし、これは、その CA が発行する **すべて** の証明書に適用されます。セキュリティーの問題が発生する可能性があるため、実稼働環境でこの設定は推奨されません。

3.7. サブジェクト名およびサブジェクト代替名の管理

証明書の **サブジェクト名** は、証明書を発行するエンティティーの ID 情報が含まれる識別名 (DN) です。このサブジェクト名は、共通名や組織単位などの標準の LDAP ディレクトリーコンポーネントから構築できます。これらのコンポーネントは X.500 に定義されます。サブジェクト名の他に、証明書には **サブジェクト代替名** があります。これは、X.500 に定義されていない追加情報が含まれる証明書の拡張機能セットです。

サブジェクト名とサブジェクトの代替名の命名コンポーネントはカスタマイズできます。



重要

サブジェクト名が空の場合は、Subject Alternative Name 拡張が存在し、critical のマークが付けられている必要があります。

3.7.1. サブジェクト名でのリクエスター CN または UID の使用

証明書要求の ***cn*** 値または ***uid*** 値は、発行した証明書のサブジェクト名をビルドするために使用できます。このセクションでは、サブジェクト名制約に naming 属性 (CN または UID) が証明書要求に存在する必要があるプロファイルを示しています。naming 属性がないと、リクエストは拒否されます。

この設定には、以下の 2 つの部分があります。

- CN または UID 形式は、Subject Name Constraint の **pattern** 設定に設定されます。
- CN または UID トークン、および証明書の特定の接尾辞を含むサブジェクト DN の形式は、Subject Name Default に設定されます。

たとえば、サブジェクト DN で CN を使用するには、次のコマンドを実行します。

```
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
```

この例では、リクエストに **cn=John Smith** の CN が含まれる場合、証明書は、 **cn=John Smith,DC=example, DC=com** のサブジェクト DN で発行されます。要求が完了しても、 **uid=jsmith** の UID があり CN がない場合、要求は拒否されます。

同じ設定を使用して、要求側の UID をサブジェクト DN にプルします。

```
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 パラメーターの形式は、 「Subject Name 制約」 および 「サブジェクト名のデフォルト」 で説明されています。

3.7.2. サブジェクト代替名への LDAP ディレクトリー属性値およびその他の情報の挿入

LDAP ディレクトリーからの情報、または要求元によって送信された情報は、Subject Alt Name Extension Default 設定で一致する変数を使用して、証明書のサブジェクト代替名に挿入できます。デフォルトでは、情報のタイプ (形式) と、情報の取得に使用する一致するパターン (変数) を設定します。以下に例を示します。

```
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
```

これにより、要求側の電子メールがサブジェクト名の最初の CN コンポーネントとして挿入されます。追加のコンポーネントを使用するには、**Type_**、**Pattern_**、および **Enable_** 値を、**Type_1** などの数値にインクリメントします。

サブジェクトの alt 名の設定については、「サブジェクト代替名の拡張機能のデフォルト」 を参照してください。

LDAP コンポーネントを証明書のサブジェクト代替名に挿入するには、以下を実行します。

1. LDAP 属性値を挿入するには、ユーザーディレクトリーの認証プラグイン **SharedSecret** を有効にする必要があります。

   a. CA コンソールを開きます。

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

   b. 左側のナビゲーションツリーで **認証** を選択します。

   c. **Authentication Instance** タブで、**Add** をクリックして、**SharedSecret** 認証プラグインのインスタンスを追加します。

   d. 以下の情報を入力します。

   ```
   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. 新規プラグインインスタンスを保存します。

   JOIN 共有トークンの設定に関する詳細は、「CMC 共有シークレットの設定」 を参照してください。

2. **ldapStringAttributes** パラメーターは、ユーザーの LDAP エントリーから ***mail*** 属性の値を読み込み、その値を証明書要求に追加するように、認証プラグインに指示します。値が要求に設定されている場合、証明書プロファイルポリシーは、拡張値のその値を挿入するように設定できます。

   dnpattern パラメーターの形式は、「Subject Name 制約」 および 「サブジェクト名のデフォルト」 で説明されています。

3. CA が証明書拡張機能に LDAP 属性の値を挿入できるようにするには、プロファイルの設定ファイルを編集し、拡張機能のポリシーセットパラメーターを挿入します。たとえば、**caFullCMCSharedTokenCert** プロファイルの Subject Alternative Name 拡張に ***mail*** 属性値を挿入するには、以下のコードを変更します。

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

   プロファイルの編集に関する詳細は、「RAW 形式での証明書プロファイルの編集」 を参照してください。

4. CA を再起動します。

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

この例では、**caFullCMCSharedTokenCert** プロファイル登録フォームを介して送信される証明書では、要求側の ***mail*** LDAP 属性の値で Subject Alternative Name 拡張機能が追加されます。以下に例を示します。

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

このポリシーセットのいずれかの **Pattern_** パラメーターにトークン (**X**) として設定することにより、証明書に自動的に挿入できる属性が多数あります。一般的なトークンは 表3.1「証明書の設定に使用する変数」 に一覧表示され、デフォルトのプロファイルにはこれらのトークンの使用方法の例が含まれています。

表3.1 証明書の設定に使用する変数

ポリシーセットトークン	説明
$request.auth_token.cn[0]$	証明書を要求したユーザーの LDAP 共通名 (***cn***) 属性。
$request.auth_token.mail[0]$	証明書を更新したユーザーの LDAP メール (***mail***) 属性の値。
$request.auth_token.tokencertsubject$	証明書サブジェクト名。
$request.auth_token.uid$	証明書を要求したユーザーの LDAP ユーザー ID (***uid***) 属性。
$request.auth_token.userdn$	証明書を要求したユーザーのユーザー DN。
$request.auth_token.userid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.uid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.requestor_email$	要求を送信したユーザーのメールアドレス。
$request.request_name$	要求を送信した人。
$request.upn$	Microsoft UPN。これには **(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.upn$** の形式があります。

ポリシーセットトークン	説明
$server.source$	サーバーに対し、サブジェクト名のバージョン 4 の UUID (乱数) コンポーネントを生成するように指示します。この値は常に **(IA5String)1.2.3.4,$server.source$** 形式になります。
$request.auth_token.user$	要求が TPS によって送信された場合に使用します。証明書をリクエストした TPS サブシステム信頼マネージャー。
$request.subject$	要求が TPS によって送信された場合に使用します。TP S が解決して要求したエンティティーのサブジェクト名 DN。例: **cn=John.Smith.123456789,o=TMS Org**

3.7.3. SAN 拡張での CN 属性の使用

RFC 2818 で非推奨となったドメイン名の検証に、サブジェクト DN の Common Name (CN) 属性の使用に対応しなくなりました。代わりに、これらのアプリケーションやライブラリーは、証明書要求で ***dNSName*** Subject Alternative Name (SAN) の値を使用します。

Certificate System は、RFC 1034 セクション 3.5 に従って優先名前構文と一致し、複数のコンポーネントを持つ場合にのみ CN をコピーします。また、既存の SAN 値が保持されます。たとえば、CN をベースとする ***dNSName*** 値は、既存の SAN に追加されます。

SAN 拡張の CN 属性を使用するように Certificate System を設定するには、証明書を発行するために使用する証明書プロファイルを編集します。以下に例を示します。

1. プロファイルを無効にします。

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

2. プロファイルを編集します。

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

   a. プロファイルに固有のセット番号を使用して、以下の設定を追加します。以下に例を示します。

```
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
```

   前述の例では、**12** をセット番号として使用しています。

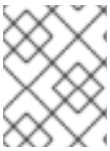
   b. ***policyset.userCertSet.list*** パラメーターに新しいポリシーセット番号を追加します。以下に例を示します。

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

c. プロファイルを保存します。

3. プロファイルを有効にします。

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



注記

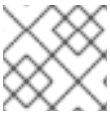
すべてのデフォルトサーバーグループに、***commonNameToSANDefaultImpl*** のデフォルトが含まれます。

3.7.4. CSR からの SAN 拡張の許可

特定の環境では、管理者は Certificate Signing Request (CSR) で Subject Alternative Name (SAN) 拡張機能を指定できるようにします。

3.7.4.1. CSR から SAN を取得するプロファイルの設定

CSR からの SAN の取得を許可するには、ユーザー拡張機能のデフォルトを使用します。詳細は、「User Supplied Extension Default」 を参照してください。



注記

SAN 拡張には、1 つ以上の SAN を含めることができます。

CSR から SAN を受け入れるには、**caCMCECserverCert** のように、以下のデフォルトおよび制約をプロファイルに追加します。

```
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. SAN を使用した CSR の生成

たとえば、**certutil** ユーティリティーを使用して、2 つの SAN を持つ CSR を生成するには、以下を実行します。

```
# 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
```

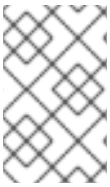
CSR の生成後に、「CMC 登録プロセス」 で説明されている手順に従って、CMC の登録を完了します。

第4章 キーアーカイブおよびリカバリーの設定

この章では、以前は Data Recovery Manager (DRM) と呼ばれていた Key Recovery Authority (KRA) を使用して秘密鍵をアーカイブし、これらのアーカイブされた鍵を復元して暗号化されたデータを復元する方法について説明します。

注記

サーバー側のキー生成は、TPS サブシステムを介して実行されるスマートカード登録用に提供されるオプションです。この章では、サーバー側のキー生成と TPS によって開始されるアーカイブではなく、クライアント側のキー生成によるキーのアーカイブについて説明します。



注記

Gemalto SafeNet LunaSA は、CKE - Key Export モデルでの PKI 秘密鍵抽出のみおよび非 FIPS モードでのみサポートされます。FIPS モードの LunaSA Cloning モデルおよび CKE モデルは、PKI 秘密鍵の抽出をサポートしません。

秘密鍵をアーカイブすることで、鍵を紛失した場合でも、ユーザーと情報を保護できます。情報は公開鍵によって暗号化されて保存されます。情報を復号化するには、対応する秘密鍵を使用できる必要があります。秘密鍵を紛失すると、データを取得できなくなります。ハードウェア障害が原因で、またはキーの所有者がパスワードを忘れた、またはキーが格納されているハードウェアトークンを失ったために、秘密キーが失われる可能性があります。同様に、キーの所有者がキーを提供できない場合、暗号化されたデータを取得することはできません。

注記

クローン環境では、キーのアーカイブとリカバリーを手動で設定する必要があります。

4.1. キーのアーカイブと回復について

エンドユーザーの観点からは、キーのアーカイブに必要なものは 2 つだけです。鍵を生成できる **pki** ユーティリティーなどのクライアントと、キーのアーカイブをサポートするように設定された証明書プロファイルです。**pki** ユーティリティーの詳細は、『Red Hat Certificate System Planning、Installation、and Deployment Guide 『(Common Criteria Edition)の コマンドラインインターフェイス (CLI』)』 セクションを参照してください。

4.1.1. キーアーカイブ

エンドエンティティーが秘密暗号化キーを紛失した場合、または秘密キーを使用できない場合、対応する公開キーで暗号化されたデータを読み取る前にキーを回復する必要があります。キーが生成されたときに秘密キーがアーカイブされていれば、回復が可能です。

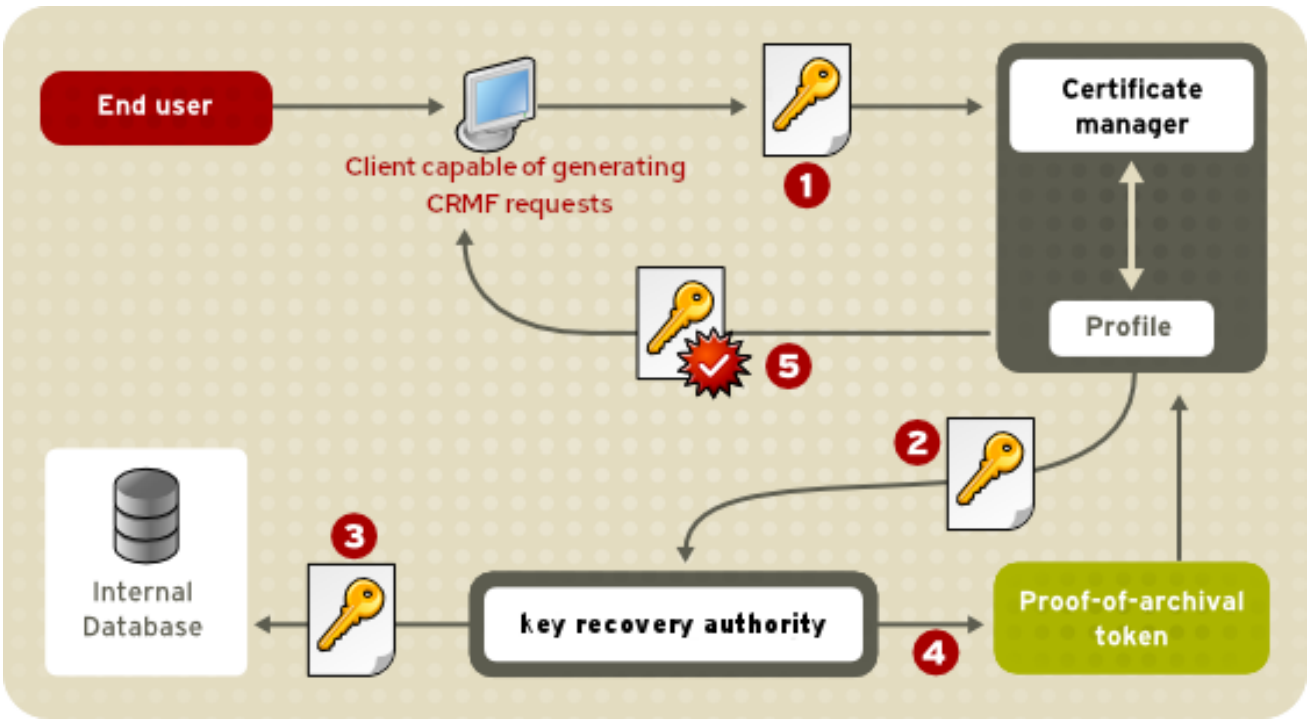
KRA は、キーレコードの形式で内部データベースの安全なキーリポジトリーに秘密暗号化キーを格納します。Certificate Manager は、要求にキーアーカイブオプションが含まれている場合、クライアントによって発行された証明書要求を KRA に自動的に転送します。このような場合、秘密鍵はクライアントの KRA トランスポートキーによってラップされ、暗号化された BLOB として KRA に送信されます。KRA は BLOB を復号化し、そのストレージキーを使用してキーを再暗号化します。次に、KRA は暗号化された BLOB に一意のキー識別子を付与し、キーレコードとしてキーリポジトリーにアーカイブします。

鍵のアーカイブコピーは、KRA のストレージキーでラップされます。ストレージ証明書の対応する秘密

鍵ペアを使用することによってのみ、復号またはラップ解除が可能になります。1 つ以上のキーリカバリー (または KRA) エージェントの証明書の組み合わせは、KRA で鍵のリカバリーを完了して秘密鍵を取得し、その証明書を使用してアーカイブされた秘密鍵を復号または再作成できるようにします。 KRA インデックスは、キー番号、所有者名、および公開鍵のハッシュ別に保存されるため、非常に効率的な検索を可能にします。

図4.1「キーアーカイブプロセスのしくみ」 は、エンドエンティティーが証明書を要求したときにキーアーカイブプロセスがどのように発生するかを示しています。

図4.1 キーアーカイブプロセスのしくみ



1. クライアントは秘密鍵を生成して暗号化し、要求を CA に送信します。

2. 証明書の要求を承認し、証明書を発行した後、Certificate Manager はそれを KRA に送信して、公開キーと共に保管します。Certificate Manager は、秘密鍵が受信および保存され、それが公開暗号化鍵に対応するという KRA からの確認を待ちます。

3. KRA は、トランスポート秘密鍵を使用して暗号化を解除します。秘密暗号化キーが公開暗号化キーに対応することを確認した後、KRA は内部データベースに保存する前に、ストレージキーの公開キーペアを使用して再度暗号化します。

4. 秘密暗号化キーが正常に保存されると、KRA はトランスポートキーペアの秘密キーを使用して、キーが正常に保存されたことを確認するトークンに署名します。次に、KRA はトークンを Certificate Manager に送信します。

5. Certificate Manager は証明書を発行します。この証明書は、CMC 応答に組み込まれています。

両方のサブシステムは、適切な段階で設定された証明書プロファイルの制約に従ってリクエストを行います。リクエストがプロファイルの制約を満たさない場合、サブシステムはリクエストを拒否します。

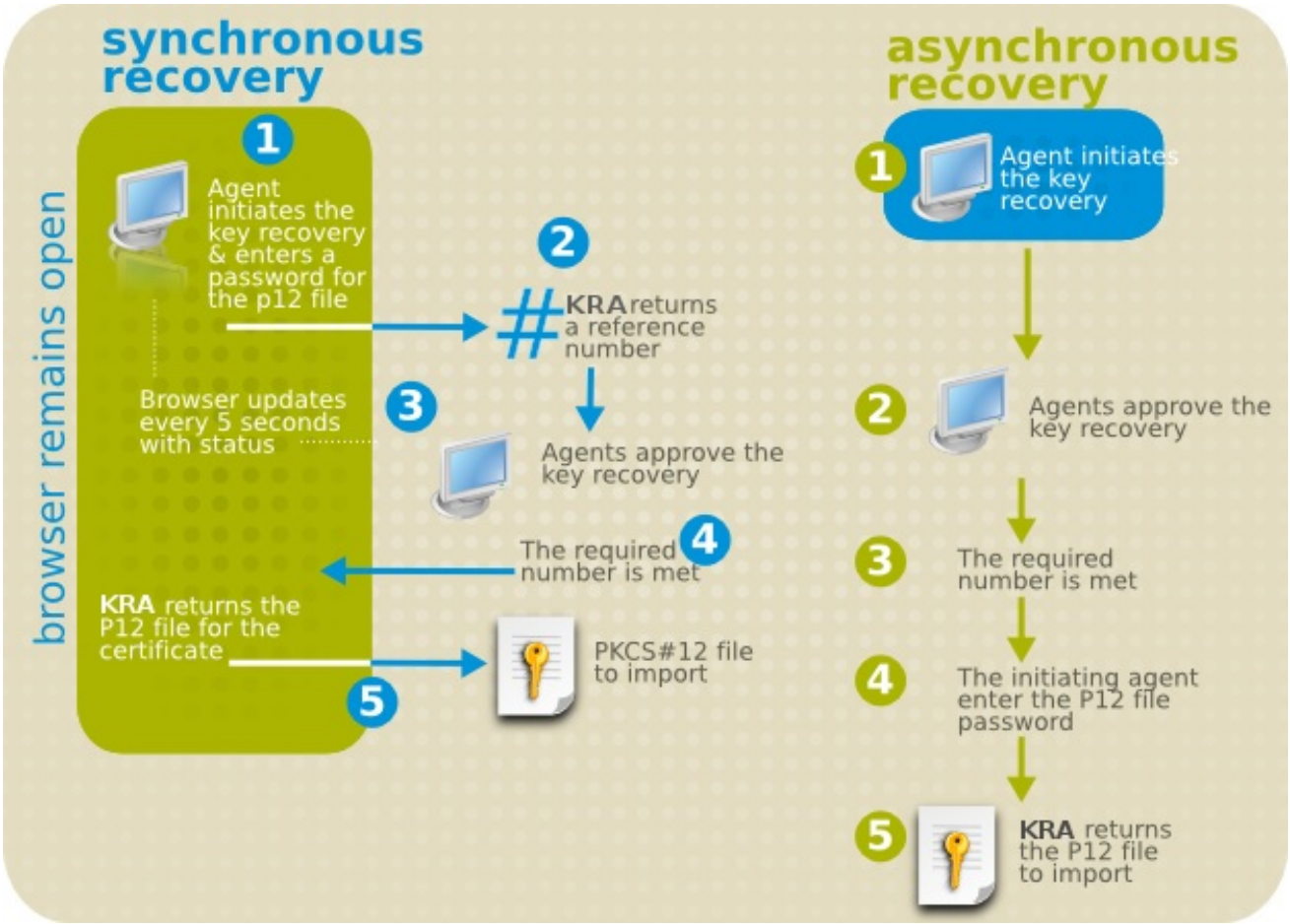
KRA は、指定された回復エージェントが KRA エージェントサービスページのキー回復フォームを使用してキー回復要求を処理および承認する場合、エージェント主導のキー回復をサポートします。特定の数のエージェントを承認すると、システムは、キーの所有者が利用できない、またはキーが失われた場合にキーを回復できます。

Certificate System 9 は、古い秘密分割 **ベースのリカバリースキームではなく、m-of-n ACL** ベースのリカバリースキーム を使用します。7.1 よりも古いバージョンの Certificate System では、ストレージトークンのパスワードが分割され、個別のリカバリーエージェントパスワードによって保護されていました。現在、Certificate System は既存のアクセス制御スキームを使用して、回復エージェントが TLS 経由で適切に認証されるようにし、エージェントが特定の回復エージェントグループ (デフォルトではキー回復機関エージェントグループ) に属していることを要求します。リカバリーリクエストは、**m-of-n** (必要な数) のリカバリーエージェントがリクエストに許可を与えた場合にのみ実行されます。

『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Setting up Agent-Approved Key Recovery Schemes』 セクションで説明されているように、KRA 設定を変更することでキー回復スキームを変更できます。

実際のキー回復プロセスでは、キー回復エージェントの 1 つが、必要なすべての回復エージェントに差し迫ったキー回復について通知し、KRA のエージェントページで回復プロセスを開始します。キーリカバリープロセスは、承認が入ったときに初期セッションを開いたままにしておく必要があることを意味する **同期** か、またはリカバリープロセスのスナップショットが内部データベースに保存され、承認が入ったときに更新されることを意味する **非同期** のいずれかです。

図4.2 非同期と同期の回復、並べて表示



重要

同期リカバリーは、Red Hat Certificate System で非推奨になりました。

非同期リカバリーの場合、各ステップは内部データベースに保存されます。エージェントは、リカバリーするキーを検索し、**Grant Recovery** ボタンをクリックしてリカバリーを承認できます。KRA が再始動されても、非同期リカバリーは持続します。

4.1.2. キーリカバリー

キーが正常にアーカイブおよびリカバリーできるかどうかをテストするには:

1. CRMF 要求を生成し、CA の登録ポータルから送信します。

   詳細は、「キーアーカイブを使用した暗号化のみの証明書の取得例」 を参照してください。

2. SMIME を実行できる電子メールクライアントに証明書をインポートします。

3. 鍵がアーカイブされたことを確認します。KRA のエージェントサービスページで、**Show completed requests** を選択します。キーが正常にアーカイブされた場合は、そのキーに関する情報が生成されます。キーが表示されない場合は、ログを確認して、問題を修正します。キーが正常にアーカイブされたら、ブラウザーウィンドウを閉じます。

4. 鍵を確認します。署名付きで暗号化された電子メールを送信します。メールを受信したら、メッセージを確認して開き、メッセージが署名されて暗号化されているかどうかを確認します。メッセージウィンドウの右上隅に、メッセージが署名されて暗号化されていることを示すセキュリティーアイコンがあるはずです。

5. 証明書を削除します。暗号化された電子メールを再度確認します。メールクライアントはメッセージを復号できないはずです。

6. アーカイブされた鍵を正常に復元できるかどうかをテストします。

   a. KRA のエージェントサービスページを開き、**Recover Keys** リンクをクリックします。キーの所有者、シリアル番号、または公開鍵で鍵を検索します。キーが正常にアーカイブされた場合は、キー情報が表示されます。

   b. **Recover** をクリックします。

   c. 表示される形式で、復元する秘密鍵に対応する base-64 でエンコードされた証明書を入力します。この情報を取得するには CA を使用します。base-64 でエンコードされた証明書を指定してアーカイブされた鍵を検索する場合は、ここで証明書を指定する必要はありません。

   d. リカバリーの実行中にブラウザーセッションが閉じられるように **Async Recovery** チェックボックスが選択されていることを確認してください。

   e. エージェントスキームに応じて、指定された数のエージェントがこの鍵のリカバリーを承認する必要があります。エージェントに、リカバリーキーを検索してもらい、開始された回復を承認してもらいます。

   f. すべてのエージェントがリカバリーを承認すると、次の画面は、証明書で PKCS #12 ファイルを暗号化するようにパスワードを要求します。

   g. 次の画面は、復元されたキーペアを含む PKCS #12 ブロブをダウンロードするリンクを返します。リンクに従い、blob をファイルに保存します。

重要

gcr-viewer ユーティリティーでブラウザーから直接 PKCS #12 ファイルを開くと、特定の状況で失敗する可能性があります。この問題を回避するには、**gcr-viewer** ファイルをダウンロードし、手動で開きます。

7. ブラウザーのデータベースに鍵を復元します。ブラウザーおよびメールクライアントに **.p12** ファイルをインポートします。

8. テストメールを開きます。メッセージは再度表示されます。

第5章 証明書の要求、登録、および管理

証明書は要求され、エンドユーザーに使用されます。証明書登録および更新は管理者に限定されていませんが、登録プロセスおよび更新プロセスを理解すると、「証明書プロファイルの設定」 で説明されているように、管理者が適切な証明書プロファイルを管理および作成でき、各証明書タイプに対して適切な認証方法を使いやすくなります。

本章では、Certificate System 外で使用する証明書の要求、受信、および更新を説明します。Certificate System サブシステム証明書の要求および更新の詳細は、14章 *サブシステム証明書の管理* を参照してください。

5.1. 証明書の登録および更新について

登録 とは、証明書の要求および受信のプロセスです。登録プロセスの仕組みは、証明書の種類、キーペアの生成方法、および証明書自体の生成と承認の方法によってわずかに異なります。特定の方法が何であれ、高レベルでの証明書の登録には、同じ基本的な手順があります。

1. ユーザーが証明書要求を生成します。
2. 証明書要求が CA に送信されます。
3. 要求は、要求したエンティティーを認証し、それを提出するために使用された証明書プロファイルのルールを満たしていることを要求が確認することで検証されます。
4. リクエストが承認されている。
5. ユーザーが新しい証明書を取得します。

証明書の有効期間が終了すると、証明書を更新できます。

5.2. 証明書署名リクエストの作成

『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『CMC を使用した登録』 セクションで説明されているように、**CMCRequest** ユーティリティーは PKCS #10 および CRMF 形式の証明書署名要求(CSR)を受け入れます。

Red Hat Certificate System は、以下のユーティリティーを使用した CSR の作成をサポートします。

- **certutil**: PKCS #10 リクエストの作成に対応します。
- **PKCS10Client**: PKCS #10 リクエストの作成に対応します。
- **CRMFPopClient**: CRMF 要求の作成をサポートします。

次のセクションでは、これらのユーティリティーを機能豊富な登録プロファイルフレームワークで使用する方法の例をいくつか示します。

5.2.1. certutil を使用した CSR の作成

本セクションでは、**certutil** ユーティリティーを使用して CSR を作成する方法を説明します。

certutil の使用の詳細は、以下を参照してください。

- certutil(1) の man ページ

- **certutil --help** コマンドの出力

5.2.1.1. certutil を使用した EC キーで CSR の作成

以下の手順は、**certutil** ユーティリティーを使用して Elliptic Curve(EC) キーペアと CSR を作成する方法を示しています。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. バイナリー CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

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

プロンプトが表示されたら、必要な NSS データベースのパスワードを入力します。

パラメーターの詳細は、certutil(1) の man ページを参照してください。

3. 作成したバイナリー形式の CSR を PEM 形式に変換します。

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

4. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/request.csr

MIICbTCCAVUCAQAwKDEQMA4GA1UEChMHRXhhbXBsZTEUMBIGA1UEAxMLZXhhbXBs

...
```

これは、PKCS#10 PEM 証明書要求です。

5. 次の手順については、「CMC 登録プロセス」 を参照してください。ただし、証明書要求の作成に関する手順はスキップしてください。

5.2.1.2. certutil を使用したユーザー定義拡張による CSR の作成

以下の手順は、**certutil** ユーティリティーを使用してユーザー定義の拡張で CSR を作成する方法を示しています。

登録要求は、CA で定義された登録プロファイルにより制限されることに注意してください。???を参照してください。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. ユーザー定義の Extended Key Usage 拡張とユーザー定義の Key Usage 拡張で CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ 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
```

   プロンプトが表示されたら、必要な NSS データベースのパスワードを入力します。

   パラメーターの詳細は、certutil(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ 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)
```

   これは、PKCS#10 PEM 証明書要求です。

4. 次の手順については、「CMC 登録プロセス」 を参照してください。ただし、証明書要求の作成に関する手順はスキップしてください。

注記

CSR からヘッダー情報を削除します。

5.2.2. PKCS10Client を使用した CSR の作成

本セクションでは、**PKCS10Client** ユーティリティーを使用して CSR を作成する方法を説明します。

PKCS10Client の使用に関する詳細は、以下を参照してください。

- PKCS10Client(1) の man ページ
- **PKCS10Client --help** コマンドの出力

5.2.2.1. PKCS10Client を使用した CSR の作成

以下の手順では、**PKCS10Client** ユーティリティーを使用して Elliptic Curve (EC) キーペアと CSR を作成する方法を説明します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

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

パラメーターの詳細は、PKCS10Client(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

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

5.2.2.2. PKCS10Client を使用した SharedSecret ベースの CMC の CSR の作成

以下の手順では、**PKCS10Client** ユーティリティーを使用して、SharedSecret ベースの CMC 用の RSA 鍵ペアと CSR を作成する方法を説明します。これは、デフォルトでは **caFullCMCSharedTokenCert** プロファイルおよび **caECFullCMCSharedTokenCert** プロファイルによって処理される CMC 共有 Secret 認証方法にのみ使用します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

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

パラメーターの詳細は、PKCS10Client(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

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

5.2.3. CRMFPopClient を使用した CSR の作成

Certificate Request Message Format (CRMF) は、CMC で受け入れられている CSR 形式であり、主要なアーカイブ情報を要求に安全に埋め込むことができます。

本セクションでは、**CRMFPopClient** ユーティリティーを使用して CSR を作成する方法を説明します。

CRMFPopClient の使用に関する詳細は、CRMFPopClient(1) の man ページを参照してください。

5.2.3.1. CRMFPopClient を使用したキー Archival を持つ CSR の作成

以下の手順では、**CRMFPopClient** ユーティリティーを使用して RSA キーペアと、鍵アーカイブオプションで CSR を作成する方法を説明します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. KRA トランスポート証明書を取得します。

```
$ 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. KRA トランスポート証明書をエクスポートします。

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

4. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ 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
```

Elliptic Curve (EC) キーペアと CSR を作成するには、**-a ec -t false** オプションをコマンドに渡します。

パラメーターの詳細は、CRMFPopClient(1) の man ページを参照してください。

5. 必要に応じて、CSR ファイルが正しいことを確認します。

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

5.2.3.2. CRMFPopClient を使用した SharedSecret ベースの CMC の CSR の作成

以下の手順では、**CRMFPopClient** ユーティリティーを使用して、SharedSecret ベースの CMC 用の

RSA 鍵ペアと CSR を作成する方法を説明します。これは、デフォルトでは **caFullCMCSharedTokenCert** プロファイルおよび **caECFullCMCSharedTokenCert** プロファイルによって処理される CMC 共有 Secret 認証方法にのみ使用します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. KRA トランスポート証明書を取得します。

```
$ 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. KRA トランスポート証明書をエクスポートします。

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

4. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ 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
```

   EC キーペアと CSR を作成するには、コマンドに **-a ec -t false** オプションを渡します。

   パラメーターの詳細は、**CRMFPopClient --help** コマンドの出力を参照してください。

5. 必要に応じて、CSR ファイルが正しいことを確認します。

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

5.3. CMC を使用した証明書の要求と受信

このセクションでは、CMS (Certificate Management over CMS) を使用して証明書を登録する手順を説明します。

CMC を使用して証明書を登録する設定とワークフローの一般的な情報は、以下を参照してください。

- 『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』『の CMC の設定』 セクション。
- 『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『CMC による登録』 セクション。
- CMCRequest(1) の man ページを参照してください。
- CMCResponse(1) の man ページを参照してください。

CMC の登録は、さまざまなシナリオの要件を満たすためにさまざまな方法で可能です。「CMC 登録プロセス」 『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Enrolling with CMC』 セクションに詳細を補足します。さらに、「実用的な CMC 登録シナリオ」 セクションを使用すると、管理者はどのメカニズムをどのシナリオで使用するかを決定できます。

5.3.1. CMC 登録プロセス

CMC を使用して証明書を要求および発行するには、次の一般的な手順を使用します。

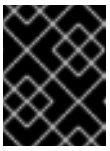
1. Certificate Signing Request (CSR) を、以下のいずれかの形式で作成します。
   - PKCS #10 形式
   - Certificate Request Message Format (CRMF) 形式:

   これらの形式で CSR を作成する方法は、「証明書署名リクエストの作成」 を参照してください。
2. 管理証明書をクライアントの NSS データベースにインポートします。以下に例を示します。
   - 以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

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

   - 11章 *証明書/キー暗号トークンの管理* のガイダンスに従って、管理クライアント証明書を検証してインポートします:

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

     重要

     CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

   - 証明書に関連付けられた秘密鍵をインポートします。

```
$ 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. 以下の内容で、**/home/user_name/cmc-request.cfg** などの CMC 要求用の設定ファイルを作成します。

```
# 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
```

詳細は、CMCRequest(1) の man ページを参照してください。

4. CMC 要求を作成します。

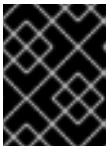
```
$ CMCRequest /home/user_name/cmc-request.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

5. **/home/user_name/cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
# 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 パラメーターで指定された証明書のニックネームは、CMC 要求で以前使用された内容と一致させる必要があります。

6. 要求する証明書のタイプに応じて、前の手順で作成した設定ファイルに次のパラメーターを追加します。

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

CA 署名証明書の場合の例を以下に示します。

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

重要

エージェントが次のステップで CMC 要求を送信する場合は、このパラメーターで指定したプロファイルは **CMCAuth** 認証プラグインを使用する必要があります。ユーザーが作成した登録では、プロファイルは **CMCUserSignedAuth** プラグインを使用する必要があります。詳細は、「CMC 認証プラグイン」 を参照してください。

7. CMC 要求を CA に送信します。

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

8. CMC の応答を PKCS #7 証明書チェーンに変換するには、**CMCResponse** ユーティリティーの ***-i*** パラメーターに CMC レスポンスファイルを渡します。以下に例を示します。

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

5.3.2. 実用的な CMC 登録シナリオ

本セクションでは、CA 管理者がどの状況でどの CMC メソッドを使用するかを決定できるようにするための、頻繁な実際の使用シナリオとそのワークフローを説明します。

CMC を使用して証明書を登録する一般的なプロセスは、「CMC 登録プロセス」 を参照してください。

5.3.2.1. システム証明書およびサーバー証明書の取得

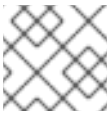
LDAP や Web サーバーなどのサービスで TLS サーバー証明書が必要な場合、このサーバーの管理者はサービスのドキュメントに基づいて CSR を作成し、承認のために CA のエージェントに送信します。このプロセスには、「CMC 登録プロセス」 で説明されている手順を使用します。また、以下の要件を考慮してください。

登録プロファイル

エージェントは、「CMC 認証プラグイン」 に記載されている既存の CMC プロファイルのいずれかを使用する必要があります。または、**CMCAuth** 認証メカニズムを使用するカスタムプロファイルを作成します。

CMC 署名証明書

システム証明書の場合、CA エージェントは CMC 要求を生成して署名する必要があります。そのためには、**CMCRequest** 設定ファイルの ***nickname*** パラメーターを CA エージェントのニックネームに設定します。



注記

CA エージェントは、独自の秘密鍵にアクセスできるようにする必要があります。

HttpClient TLS Client Nickname

HttpClient の設定ファイル内で TLS クライアント認証に関するユーティリティーの設定ファイルに対して、**CMCRequest** ユーティリティー設定ファイルへのサインインに同じ証明書を使用します。

HttpClient *servlet* パラメーター

HttpClient ユーティリティーに渡される設定ファイルの ***servlet*** では、要求を処理する CMC サーブレットおよび登録プロファイルが参照されます。

要求する証明書のタイプに応じて、直前の手順で作成した設定ファイルに以下のエントリーのいずれかを追加します。

- CA 署名証明書の場合:

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

- KRA トランスポート証明書の場合:

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

- OCSP 署名証明書の場合:

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

- 監査署名証明書の場合:

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

- サブシステム証明書の場合:
  - RSA 証明書の場合:

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

  - ECC 証明書の場合:

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

- TLS サーバー証明書の場合:
  - RSA 証明書の場合:

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

  - ECC 証明書の場合:

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

- 管理証明書の場合:

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

詳細は以下を参照してください。

- エージェントが CSR を事前署名する場合、エージェントは識別のために CSR を調べるため、Proof of Identification が確立されたと見なされます。追加の CMC 固有の識別証明は必要ありません。
- PKCS #10 ファイルはすでに Proof of Possession (POP) 情報を提供し、追加の Proof of Possession (POP) は必要ありません。
- エージェントの事前承認済みリクエストでは、識別はエージェントによってチェックされるため、**PopLinkWittnessV2** 機能を無効にする必要があります。

5.3.2.2. ユーザーの初回署名証明書の取得

ユーザーの最初の署名証明書を承認する方法は 2 つあります。

- エージェントは CMC 要求を署名します。「エージェント証明書を使用した CMC 要求の署名」を参照してください。
- 証明書の登録は、共有シークレットを使用して認証されます。「共有シークレットを使用した証明書の登録の認証」 を参照してください。

5.3.2.2.1. エージェント証明書を使用した CMC 要求の署名

エージェント証明書を使用して CMC 要求に署名するプロセスは、「システム証明書およびサーバー証明書の取得」 で説明されているシステム証明書およびサーバー証明書の場合と同じです。唯一の違いは、ユーザーが CSR を作成し、承認のために CA エージェントに送信することです。

5.3.2.2.2. 共有シークレットを使用した証明書の登録の認証

ユーザーが最初の署名証明書を取得したいが、エージェントが、「エージェント証明書を使用した CMC 要求の署名」 で説明されているように要求を承認できない場合は、共有トークンを使用できます。このトークンを使用すると、ユーザーは最初の署名証明書を取得できます。次に、この証明書を使用してユーザーの他の証明書に署名できます。

このシナリオでは、Shared Secret のメカニズムを使用して、ユーザーの最初の署名証明書を取得します。「CMC 登録プロセス」 とともに以下の情報を使用します。

1. ユーザーまたは CA 管理者として共有トークンを作成します。詳細は、「共有シークレットトークンの作成」 を参照してください。

   以下の点に留意してください。

   - ユーザーがトークンを作成した場合、ユーザーはトークンを CA 管理者に送信する必要があります。
   - CA 管理者がトークンを作成した場合、管理者はユーザーがトークンを生成するのに使用するパスワードを共有する必要があります。セキュアな方法でパスワードを送信します。

2. CA 管理者として、LDAP のユーザーエントリーに Shared Token を追加します。詳細については、『Red Hat Certificate System Planning, Installation, and Deployment Guide (Common Criteria Edition)』 の 「証明書の登録用ユーザーエントリーへの CMC 共有シークレットの追加」 と 『Enabling the CMC Shared Secret Feature』 セクションを参照してください。

3. **CMCRequest** ユーティリティーに渡される設定ファイルで以下のパラメーターを使用します。

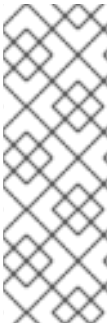
   - ***identification.enable***
   - ***witness.sharedSecret***
   - ***identityProofV2.enable***
   - ***identityProofV2.hashAlg***
   - ***identityProofV2.macAlg***
   - ***request.useSharedSecret***
   - ***request.privKeyId***

4. CA で必要な場合は、**CMCRequest** ユーティリティーに渡される設定ファイルで以下のパラメーターも使用します。

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

5.3.2.3. ユーザーの暗号化のみの証明書の取得

本セクションでは、既存のユーザー署名証明書で署名された暗号化のみの証明書を取得するワークフローを説明します。



注記

ユーザーがさまざまな用途で複数の証明書を所有していて、1 つが署名している場合、ユーザーは最初に署名証明書を取得する必要があります。ユーザーが署名証明書を所有すると、CMC Shared Secret メカニズムを設定して依存することなく、Proof Of Origin に使用できます。

ユーザーの最初の署名証明書を取得する方法は、「ユーザーの初回署名証明書の取得」を参照してください。

ユーザーとして以下を行います。

1. Network Security Services (NSS) データベースまたはユーザーの署名証明書および鍵が含まれるスマートカードに保存されている暗号化トークンを使用します。
2. PKCS #10 形式または CRMF 形式で CSR を生成します。

   注記

   (キーのアーカイブが必要な場合は) CRMF 形式を使用してください。

3. CMC 要求を生成します。

   これは暗号のみの証明書であるため、秘密鍵は署名できません。そのため、Proof Of Possession (POP) は含まれていません。このため、登録には、2 つの手順が必要です。最初のリクエストが成功すると、**EncryptedPOP** 制御のある CMC 状態が生じます。次に、ユーザーは応答を使用して、**DecryptedPOP** 制御を含む CMC 要求を生成し、2 番目のステップで送信します。

   a. 最初のステップでは、デフォルトのパラメーターに加えて、ユーザーは、**CMCRequest** ユーティリティーに渡される設定ファイルに次のパラメーターを設定する必要があります。

      - ***identification.enable***
      - ***witness.sharedSecret***
      - ***identityProofV2.enable***
      - ***identityProofV2.hashAlg***
      - ***identityProofV2.macAlg***
      - ***popLinkWitnessV2.enable*** (CA で必要な場合)
      - ***popLinkWitnessV2.keyGenAlg*** (CA で必要な場合)
      - ***popLinkWitnessV2.macAlg*** (CA で必要な場合)
      - ***request.privKeyId***

      詳細は、CMCRequest(1) の man ページを参照してください。

応答には以下が含まれます。

- CMC で暗号化された POP コントロール
- **POP の required** エラーでの **CMCStatusInfoV2** コントロール
- リクエスト ID

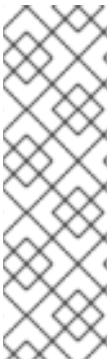
b. 次のステップでは、デフォルトのパラメーターに加えて、ユーザーは、**CMCRequest** ユーティリティーに渡される設定ファイルに次のパラメーターを設定する必要があります。

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

詳細は、CMCRequest(1) の man ページを参照してください。

5.3.2.3.1. キーアーカイブを使用した暗号化のみの証明書の取得例

キーアーカイブを使用して登録を実行するには、CRMF 要求にユーザーの暗号化された秘密鍵を含む CMC 要求を生成します。以下の手順は、ユーザーが署名証明書をすでに所有していることを前提としています。この署名証明書のニックネームは、手順の設定ファイルに設定されます。



注記

以下の手順は、署名に使用できない暗号のみの鍵で使用される 2 通の発行を説明します。証明書に署名できるキーを使用する場合は、**-q POP_NONE** の代わりに **-q POP_SUCCESS** オプションを、単一トリップ発行のために **CRMFPopClient** ユーティリティーに渡します。

POP_SUCCESS で **CRMFPoPClient** を使用する方法は、「**CRMFPopClient** を使用したキー Archival を持つ CSR の作成」 および 「**CRMFPopClient** を使用した SharedSecret ベースの CMC の CSR の作成」 を参照してください。

1. KRA トランスポート証明書を検索します。以下に例を示します。

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

2. 前の手順で取得した KRA トランスポート証明書のシリアル番号を使用して、証明書をファイルに保存します。たとえば、**/home/user_name/kra.cert** ファイルに *12345* シリアル番号がある証明書を保存するには、次のコマンドを実行します。

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

3. **CRMFPopClient** ユーティリティーを使用して以下を行います。

- キーアーカイブを使用して CSR を作成します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

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

2. RSA 秘密鍵が KRA トランスポート証明書によりラップされる CRMF 要求を作成するには、**CRMFPopClient** ユーティリティーを使用します。たとえば、要求を **/home/user_name/crmf.req** ファイルに保存するには、以下のコマンドを実行します。

```
$ 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
```

コマンドで表示される秘密鍵の ID をメモします。ID は、2 番目のトリップの設定ファイルの ***request.privKeyId*** パラメーターの値として、後のステップで必要になります。

4. 以下の内容を含む、**/home/*user_name*/cmc.cfg** など、**CRMRequest** ユーティリティー用の設定ファイルを作成します。

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

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

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

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

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

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

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

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

5. CMC 要求を作成します。

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

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

6. **/home/user_name/cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

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

#port: port number
port=8443

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

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

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

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

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

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

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

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

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

7. CMC 要求を CA に送信します。

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

コマンドが成功すると、**HTTPClient** ユーティリティーは、CMC 応答を、設定ファイルの ***output*** パラメーターで指定されたファイルに保存します。

8. 応答ファイルを **CMCResponse** ユーティリティーに渡して応答を確認します。以下に例を示します。

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

最初のトリップが成功した場合は、**CMCResponse** は、以下のような出力を表示します。

```
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. 2 番目のトリップの場合は、後の手順で使用する **/home/user_name/cmc_DecryptedPOP.cfg** などの **DecryptedPOP** の設定ファイルを作成します。作成されたファイルに以下の内容を追加します。

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

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

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

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

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

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

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

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

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

10. **DecryptPOP** CMC 要求を作成します。

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

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***decryptedPopRequestFile*** パラメーターで指定されたファイルに CMC 要求を保存します。

11. **/home/user_name/decrypted_POP_cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で **DecryptedPOP** CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

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

#port: port number
port=8443

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

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

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

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

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

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

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

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

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

12. **DecryptedPOP** CMC 要求を CA に送信します。

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

コマンドが成功すると、**HTTPClient** ユーティリティーは、CMC 応答を、設定ファイルの ***output*** パラメーターで指定されたファイルに保存します。

13. CMC の応答を PKCS #7 証明書チェーンに変換するには、**CMCResponse** ユーティリティーの ***-i*** パラメーターに CMC レスポンスファイルを渡します。以下に例を示します。

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

または、個々の証明書を PEM 形式で表示するには、**-v** ユーティリティーに渡します。

次のトリップが成功した場合は、**CMCResponse** は、以下のような出力を表示します。

```
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
```

5.4. 証明書の更新

このセクションでは、「更新について」 で説明されているさまざまなタイプの証明書更新の使用方法について説明します。このセクションで説明する方法を使用して、エージェントの承認がある場合とない場合の両方で証明書を更新できます。エージェントの承認なしでユーザーとして証明書を更新するには、**CMCUserSignedAuth** 認証プラグインを必要とするプロファイルを使用し、エージェントの承認で更新するには、**CMCAuth** 認証プラグインを必要とするプロファイルを使用します。これらのプラグイン、およびデフォルトで有効にされているプロファイルの詳細は、「CMC 認証プラグイン」 を参照してください。

5.4.1. 同じ鍵を使用した更新

「CMC 登録プロセス」 では、CMC を使用して証明書を要求および発行する方法を説明します。ユーザーが同じ登録プロファイルを使用してこのプロセス中に作成された同じ CMC 要求を送信すると、Certificate System は同じキーで証明書を更新します。

注記

同じキーを使用してユーザーとして証明書を更新する場合は、「更新について」 および「同じ鍵を使用した更新」 で説明されているように、登録プロファイルに ***params.allowSameKeyRenewal*** パラメーターが **True** に設定された ***uniqueKeyConstraint*** エントリーが含まれている必要があります。

5.4.2. 新しい鍵を使用した更新

新しいキーを使用して証明書を更新するには、「CMC 登録プロセス」 で説明されている手順に従います。更新のプロセスは、新しい登録と同じです。同じ署名証明書で要求に署名すると、新たに発行された証明書には、署名証明書と同じ ***subjectDN*** 属性が含まれます。

5.5. 発行された証明書の CSR へのトレース、および発行された証明書の CSR へのトレース

本セクションでは、CA エージェントが、元の送信された CSR への発行済み証明書を追跡し、CSR から発行された証明書を追跡する方法を説明します。

証明書要求が正常に承認されると、CA エージェントは以下を実行して要求を検索し、証明書に一致する CSR を確認できます。

1. **https://*host_name*:*port*/ca/agent/ca** にアクセスします。
2. **Search for Requests**をクリックします。
3. **Request ID Range** を選択して入力します（例： **12**（最低リクエスト **ID の場合は 12、Highest Request ID** の場合は 12）。
4. **Request Type** を選択し、**登録** タイプを選択します。
5. **Request Status** を選択し、**completed** status を選択します。
6. その他すべてが選択されていないことを確認してください。
7. **送信** をクリックします。
8. 要求番号をクリックします。この時点で証明書はクリアテキストで表示されます。
9. CSR と証明書のリンクを表示するには、右クリックして **This Frame** and **View Frame Source** を選択します。
    - **inputList.inputName="Certificate Request";** を検索します。リクエストは、それに続く **inputList.inputVal** です。
    - **outputList.outputSyntax="pretty_print";** を検索します。証明書は、それに続く **outputList.outputVal** です。

証明書から CSR を検索するには、以下を実行します。

1. **https://*host_name*:*port*/ca/agent/ca** にアクセスします。

2. **Find** をクリックします。
3. **Details** をクリックします。
4. 証明書はクリアテキストで、要求 ID リンクと共に表示されます。リンクをクリックし、**Request** ページを開きます。
5. 証明書および CSR リンクを表示するには、以下を実行します。
   - **inputList.inputName="Certificate Request";** を検索します。リクエストは、それに続く **inputList.inputVal** です。
   - **outputList.outputSyntax="pretty_print";** を検索します。証明書は、それに続く **outputList.outputVal** です。

第6章 証明書の取り消しおよび CRL 発行

Certificate System は、証明書の取り消しと、失効した証明書一覧 (CRL) と呼ばれる失効証明書の一覧を生成する方法を提供します。この章では、証明書を取り消す方法と、CMC の取り消しについて説明し、CRL と CRL 設定について詳しく説明します。

6.1. 証明書の失効について

証明書は、エンドユーザー (証明書の元の所有者) または Certificate Manager エージェントによって取り消すことができます。エンドユーザーは、認証用に提示された証明書と同じサブジェクト名が含まれる証明書のみを取り消します。

失効要求が承認されると、Certificate Manager は、その内部データベースで対応する証明書レコードを失効し、これを設定すると、公開ディレクトリーから失効した証明書が削除されます。これらの変更は、CA が発行する次の CRL に反映されます。

ID トークンとして公開鍵証明書を使用するサーバーおよびクライアントアプリケーションには、証明書の有効性に関する情報へのアクセスが必要です。証明書の有効性を決定する要素の 1 つが失効ステータスであるため、これらのアプリケーションは、検証する証明書が取り消されているかどうかを確認する必要があります。CA は以下を行う責任があります。

- 失効要求が CA によって受け取られ、承認されている場合は、証明書を取り消します。
- 失効した証明書のステータスを、その有効性ステータスを確認する必要のある関係者またはアプリケーションが利用できるようにします。

証明書が取り消されるたびに、Certificate Manager は内部データベース内の証明書のステータスを自動的に更新し、内部データベース内の証明書のコピーを失効としてマークし、データベースから証明書を削除するように Certificate Manager が設定されている場合は、失効した証明書を公開ディレクトリーから削除します。

証明書の失効ステータスを通信するための標準的な方法の 1 つは、失効した証明書のリスト (証明書失効リスト (CRL)) を公開することです。CRL は、失効した証明書の公開されている証明書の公開されているリストです。

Certificate Manager は CRL を生成するように設定できます。これらの CRL は、CRL 設定で拡張固有のモジュールを有効にすることで、X.509 標準に準拠するように作成できます。サーバーは、CRL 発行ポイントフレームワークを介して標準の CRL 拡張をサポートします。発行ポイントの CRL 拡張設定の詳細は、「CRL 拡張機能の設定」 を参照してください。証明書マネージャーは、証明書が取り消されるたびに、定期的に CRL を生成できます。公開が設定されている場合、CRL はファイル、LDAP ディレクトリー、または OCSP レスポンダーに公開できます。

CRL は、CRL にリストされている証明書を発行した CA によって、またはその CA によって CRL の発行を許可されたエンティティーによって発行され、デジタル署名されます。CA は、単一のキーペアを使用して、発行する証明書と CRL の両方に署名することも、2 つのキーペアを、1 つは発行する証明書、もう 1 つは CRL にそれぞれ使用することもできます。

デフォルトでは、Certificate Manager は 1 つのキーペアを使用して、発行する証明書を署名し、生成する CRL を生成します。Certificate Manager 用に別のキーペアを作成し、それを CRL の署名専用に使用するには、9.2.3.11 を参照してください。Red Hat Certificate System の Planning、Installation、and Deployment Guide の別の証明書を使用して CRL に署名するように CA を設定する。

CRL は、発行ポイントが定義および設定されているとき、および CRL 生成が有効になっているときに生成されます。

CRL が有効になっている場合、サーバーは証明書が取り消されるときに失効情報を収集します。サー

バーは、設定されたすべての発行ポイントに対して、取り消された証明書との一致を試みます。特定の証明書は、どの発行ポイントとも一致できないか、1つの発行ポイント、複数の発行ポイント、またはすべての発行ポイントと一致します。取り消された証明書が発行ポイントと一致すると、サーバーは証明書に関する情報をその発行ポイントのキャッシュに格納します。

キャッシュをコピーする間隔は秒単位で内部ディレクトリーにコピーされます。CRL を作成する間隔に達すると、CRL がキャッシュから作成されます。この発行ポイントにデルタ CRL が設定されている場合は、この時点でデルタ CRL も作成されます。Certificate Manager がこの情報の収集を開始したため、完全な CRL には、失効した証明書情報がすべて含まれます。デルタ CRL には、完全な CRL の最終更新以降、取り消されたすべての証明書情報が含まれます。

完全な CRL は、デルタ CRL のように順番に番号が付けられます。完全な CRL とデルタは同じ番号を持つことができます。この場合、デルタ CRL は **次** の完全な CRL と同じ番号を持ちます。たとえば、完全な CRL が最初の CRL の場合、これは CRL 1 になります。デルタ CRL は Delta CRL 2 です。CRL 1 と Delta CRL 2 で結合されたデータは、次の完全な CRL である CRL 2 と同等です。

注記

発行ポイントの拡張に変更を加えると、その発行ポイントに対して次の完全な CRL でデルタ CRL が作成されません。デルタ CRL は、作成される **2 番目** の完全な CRL で作成され、その後のすべての完全な CRL と共に作成されます。

内部データベースには、最新の CRL および delta CRL のみが保存されます。新しい CRL が作成されると、古い CRL が上書きされます。

CRL が公開されると、CRL およびデルタ CRL の各更新は、公開設定で指定された場所に公開されます。公開する方法は、保存される CRL の数を決定します。ファイル公開では、各 CRL は、CRL の番号を使用してファイルに公開されるため、ファイルは上書きされません。LDAP 公開では、公開される各 CRL はディレクトリーエントリーに CRL を含む属性の古い CRL に置き換えられます。

デフォルトでは、CRL には失効した証明書に関する情報が含まれません。サーバーには、発行ポイントにオプションを有効にして、失効した証明書を含めることができます。期限切れの証明書が含まれている場合、失効した証明書に関する情報は、証明書の有効期限が切れても CRL から削除されません。失効した証明書が含まれていない場合は、証明書の有効期限が切れると、失効した証明書に関する情報が CRL から削除されます。

6.1.1. CRL 発行ポイント

CRL は非常に大きくなる可能性があるため、大きな CRL の取得と配信のオーバーヘッドを最小限に抑える方法は複数あります。この方法の 1 つは、証明書領域全体を分割し、個別の CRL を各パーティションに関連付けます。このパーティションは *CRL 発行ポイント* と呼ばれます。これは、失効した全証明書のサブセットが保持される場所です。パーティション設定は、取り消された証明書が CA 証明書であるかどうか、特定の理由で取り消されたかどうか、または特定のプロファイルを使用して発行されたかどうかに基づいて行うことができます。各発行ポイントは名前で識別されます。

デフォルトでは、Certificate Manager は単一の CRL (*マスター CRL*) を生成し、公開します。発行ポイントは、すべての証明書、CA 署名証明書のみ、または期限切れの証明書を含むすべての証明書の CRL を生成できます。

発行ポイントを定義したら、それらを証明書に含めることができるため、証明書の失効ステータスを確認する必要があるアプリケーションは、マスターまたはメイン CRL の代わりに、証明書で指定された CRL 発行ポイントにアクセスできます。発行ポイントで維持される CRL はマスター CRL よりも小さいため、失効ステータスの確認ははるかに高速です。

CRL データベースパブリケーションポイントは、**CRLDistributionPoint** 拡張機能を設定して証明書に関連付

CRL ディストリビューションポイントは、**GRLDistributionPoint** 拡張機能を設定して証明書に関連付けることができます。

6.1.2. Delta CRL

デルタ CRL は、定義された発行ポイントに対して発行できます。デルタ CRL には、完全な CRL への最後の更新以降に取り消された証明書に関する情報が含まれます。発行先の Delta CRL は、**DeltaCRLIndicator** 拡張を有効にして作成されます。

6.1.3. CRL の公開

Certificate Manager は CRL をファイル、LDAP 準拠のディレクトリー、または OCSP レスポンダーに公開できます。CRL が公開される場所と頻度は、*7章証明書および CRL の公開* で説明されているように、証明書マネージャーで設定されます。

CRL は非常に大きくなる可能性があるため、CRL の公開には非常に長い時間がかかる可能性があり、プロセスが中断される可能性があります。特別なパブリッシャーは、HTTP 1.1 を介して CRL をファイルに発行するように設定できます。プロセスが中断された場合、CA サブシステムの Web サーバーは、最初からではなく、中断された時点から発行を再開できます。この操作は、「再開可能な CRL ダウンロードの設定」 に説明があります。

6.2. 証明書の取り消し

6.2.1. CMC 失効の実行

CMS (CMC) 登録を介した Certificate Management と同様に、CMC 失効により、ユーザーは失効クライアントをセットアップし、一致する **subjectDN** 属性を使用するエージェント証明書またはユーザー証明書のいずれかを使用して失効要求に署名できます。これにより、ユーザーは署名済み要求を Certificate Manager に送信できます。

または、Shared Secret Token メカニズムを使用して CMC の失効を認証することもできます。詳細は、「CMC SharedSecret 認証」 を参照してください。

ユーザーまたはエージェントが要求に署名するかどうか、または Shared Secret Token が使用されているかどうかに関係なく、Certificate Manager は、有効な失効要求を受信すると、証明書を自動的に失効させます。

Certificate System は、CMC 失効要求用に以下のユーティリティーを提供します。

- **CMCRequest**詳細は、「**CMCRequest** を使用した証明書の取り消し」 を参照してください。
- **CMCRevoke**詳細は、「**CMCRevoke**を使用した証明書の取り消し」 を参照してください。

重要

Red Hat は、**CMCRequest** ユーティリティーを使用して、失効要求の生成を推奨します。これは、**CMCRevoke** よりも多くのオプションを提供するためです。

6.2.1.1. CMCRequest を使用した証明書の取り消し

CMCRequest を使用して証明書を取り消すには、以下を実行します。

1. 以下の内容で、**/home/user_name/cmc-request.cfg** などの CMC 失効要求の設定ファイルを作成します。

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

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

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

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

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

#password: password for cert8.db which stores the user signing
#certificate and keys
password=myPass

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

## revocation parameters
revRequest.enable=true
revRequest.serial=45
revRequest.reason=unspecified
revRequest.comment=user test revocation
revRequest.issuer=issuer
revRequest.sharedSecret=shared_secret
```

2. CMC 要求を作成します。

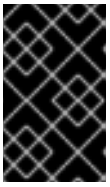
```
# CMCRequest /home/user_name/cmc-request.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

3. **/home/user_name/cmc-submit.cfg** などの設定ファイルを作成します。このファイルは、後で CMC 取り消し要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
#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 TLS client authentication certificate
#can be found (default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

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

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

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

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



重要

CMC 失効要求に署名されている場合は、***secure*** パラメーターおよび ***clientmode*** パラメーターも **true** に設定します。さらに ***nickname*** パラメーターも入力します。

4. 要求に署名したユーザーに応じて、**HttpClient** の設定ファイルの ***servlet*** パラメーターを適切に設定する必要があります。
   - エージェントが要求に署名した場合は、以下を設定します。

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

   - ユーザーが要求に署名した場合は、以下を設定します。

     ```
     servlet=/ca/ee/ca/profileSubmitUserSignedCMCFull
     ```

5. CMC 要求を送信します。

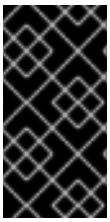
   ```
   # HttpClient /home/user_name/cmc-submit.cfg
   ```

CMCRequest を使用して証明書の取り消しに関する詳細は、CMCRequest(1) の man ページを参照してください。

6.2.1.2. CMCRevokeを使用した証明書の取り消し

CMC 失効ユーティリティー **CMCRevoke** は、エージェントの証明書を使用して失効要求に署名するために使用されます。このユーティリティーは、必要な情報 (証明書のシリアル番号、発行者名、失効理由) を渡して取り消す証明書を識別し、次に失効を実行する CA エージェントを識別するための必要な

情報 (証明書のニックネームと証明書のあるデータベース) を渡します。



重要

CMCRevoke を使用するには、CA 管理者が、インストール時に 『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Web ユーザーインターフェイスでの CMCRevoke の有効化』 セクションで指定されている指示に従う必要があります。

証明書が取り消される理由は、次のいずれかです (数値は、**CMCRevoke** に渡される値です)。

- **0**: 指定無し
- **1**: キーが侵害されました。
- **2**: CA キーが侵害されました。
- **3**: 従業員の所属が変更になりました
- **4**: 証明書が置き換えられました
- **5**: 運用停止
- **6**: 証明書が保留中です

利用可能なツール引数は、『コマンドラインツールツールガイド』 で詳細に説明されています。

6.2.1.2.1. CMCRevoke のテスト

1. 既存の証明書の CMC 失効要求を作成します。

```
CMCRevoke -d/path/to/agent-cert-db -nnickname -iissuerName -sserialName -mreason -
ccomment
```

たとえば、エージェント証明書を含むディレクトリーは **~jsmith/.mozilla/firefox/** で、証明書のニックネームは **AgentCert** で、証明書のシリアル番号は **22** の場合、コマンドは次のとおりです

```
CMCRevoke -d"~jsmith/.mozilla/firefox/" -n"ManagerAgentCert" -i"cn=agentAuthMgr" -s22 -
m0 -c"test comment"
```

注記

引用符で囲まれたスペースを含む値を囲みます。

重要

引数とその値の間には空白を入れないでください。たとえば、26 のシリアル番号は **-26** ではなく、**-s 26** となります。

2. エンドエンティティーを開きます。

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

3. **Revocation** タブを選択します。
4. メニューの **CMC Revoke** リンクを選択します。
5. **CMCRevoke** からテキストエリアに出力を貼り付けます。
6. 貼り付けられたコンテンツから **-----BEGIN NEW CERTIFICATE REQUEST-----** および **----END NEW CERTIFICATE REQUEST-----** を削除します。
7. **送信** をクリックします。
8. 返されるページは、正しい証明書が取り消されていることを確認します。

6.2.2. Web UI からのエージェントとしての失効の実行

Certificate Manager エージェントは、エージェントサービスページを使用して、Certificate System が発行する特定の証明書を見つけるか、指定の基準に一致する証明書の一覧を取得できます。取得する証明書は、エージェントによって検査または取り消されます。Certificate Manager エージェントは、証明書失効リスト (CRL) も管理できます。

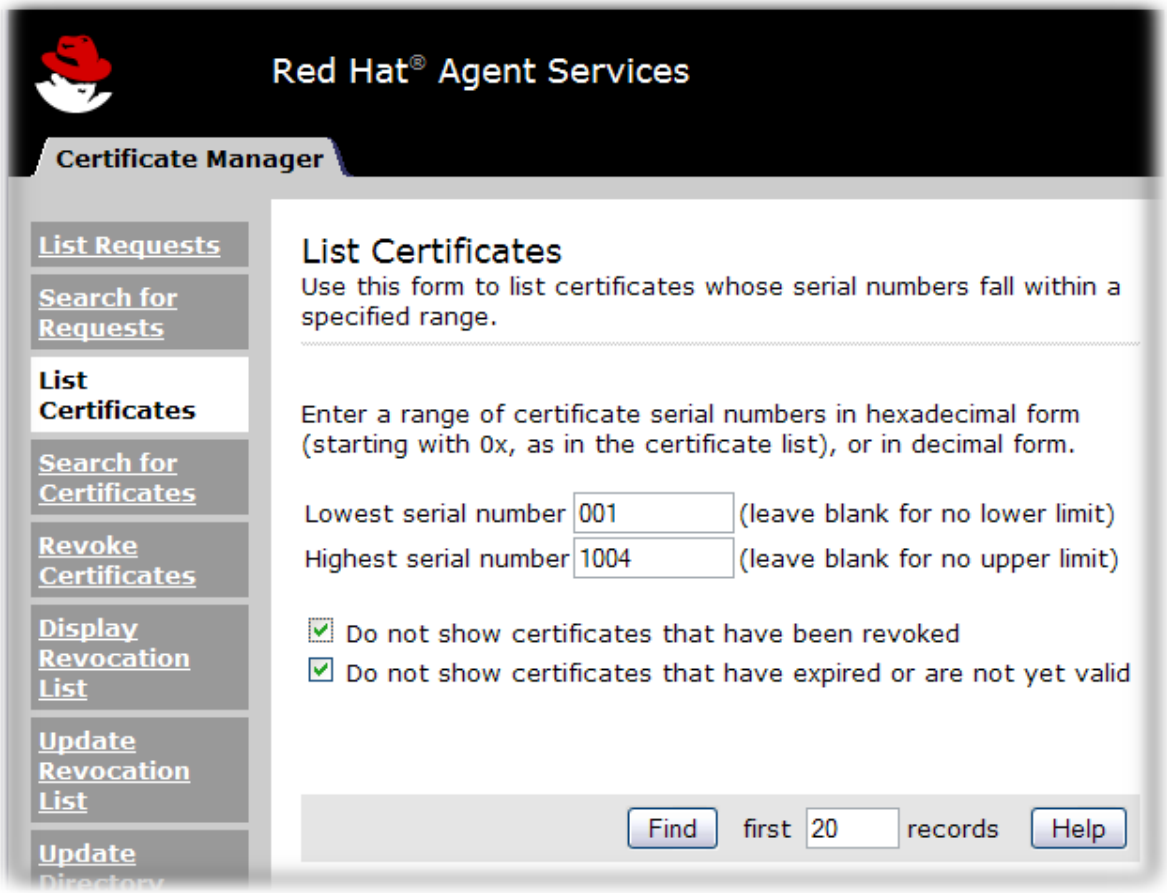
6.2.2.1. 証明書の一覧表示

シリアル番号の範囲内で証明書を一覧表示することができます。範囲内のすべての証明書が表示されるか、エージェントを選択した場合は、現在有効な証明書のみが表示されます。

特定の証明書を見つけるか、シリアル番号で証明書を一覧表示するには、以下を実行します。

1. Certificate Manager エージェントサービスページを開きます。
2. **証明書の一覧表示** をクリックします。

図6.1 証明書の一覧表示



- 特定のシリアル番号を持つ証明書を検索するには、**List Certificates** フォームの上限フィールドと下限フィールドの両方にシリアル番号を 10 進数または 16 進数形式で入力します。16 進数の先頭を指定するには **0x** を使用します（例： **0x00000006** ）。シリアル番号は、**検索結果** ページと **詳細** ページに 16 進数で表示されます。

- シリアル番号の範囲内にある証明書をすべて検索するには、シリアル番号の範囲の上限と下限を 10 進数または 16 進数の形式に入力します。

制限または上限フィールドを空白のままにすると、指定された番号の証明書と、シーケンスの前または後のすべての証明書が表示されます。

3. 返されたリストを有効な証明書に制限するには、フィルターメソッドでラベルが付けられたチェックボックスを選択します。失効した証明書を含めることも、有効ではない期限切れの証明書や証明書を追加したり、有効な証明書のみを表示したりできます。

4. results ページで返される条件に一致する証明書の最大数を入力します。

   数値を入力すると、条件に一致する番号までの最初の証明書が表示されます。

5. **Find** をクリックします。

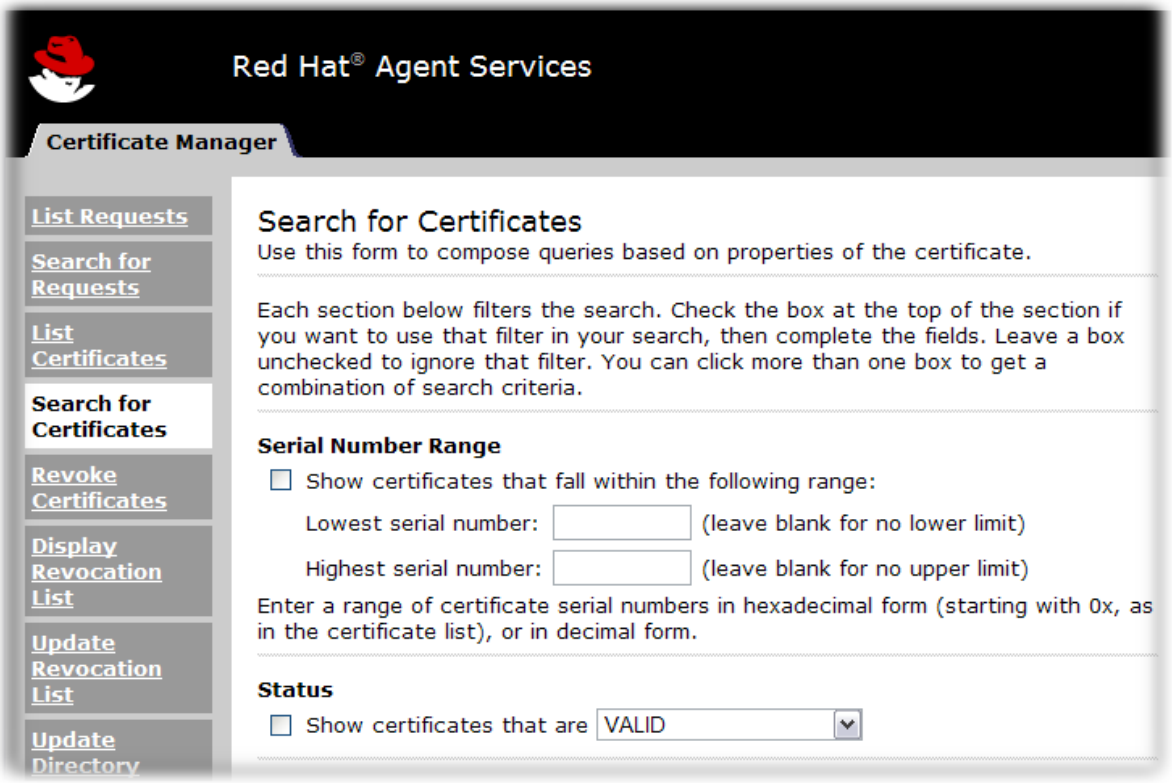
   Certificate System は、検索条件に一致する証明書の一覧を表示します。一覧内の証明書を選択して詳細を確認し、さまざまな操作を実行します。詳細は、「証明書の詳細の調査」 を参照してください。

6.2.2.2. 証明書の検索 (詳細)

高度な検索フォームを使用して、シリアル番号よりも複雑な条件で証明書を検索します。証明書の高度な検索を実行するには、以下を実行します。

1. Certificate Manager エージェントサービスページを開きます。エージェントは、このページにアクセスするために適切なクライアント証明書を送信する必要があります。

2. **Search for Certificates** をクリックし、**Search for Certificates** フォームを表示し、検索条件を指定します。



3. 特定の基準で検索するには、**Search for Certificates** フォームのセクションを 1 つ以上使用します。セクションを使用するには、チェックボックスを選択して、必要な情報を入力します。

   - Serial Number Range特定のシリアル番号を持つ証明書を見つけるか、シリアル番号の範囲内の全証明書を一覧表示します。

     - 特定のシリアル番号を持つ証明書を検索するには、最大制限フィールドと下限フィールドに、10 進数または 16 進数のいずれかでシリアル番号を入力します。 **0x 2A** などの 16 進数の先頭を指定するには、0x を使用します。シリアル番号は、**検索結果** ページと **詳細** ページに 16 進数で表示されます。

     - シリアル番号の範囲内にある証明書をすべて検索するには、シリアル番号の範囲の上限と下限を 10 進数または 16 進数でに入力します。制限または上限フィールドのいずれかを空白のままにすると、指定された数の前後にすべての証明書が返されます。

   - **状態。**ステータスで証明書を選択します。証明書には、以下のステータスコードのいずれかがあります。

     - Valid有効な証明書が発行され、有効期間は開始されましたが、終了せず、取り消されていません。

     - Invalid無効な証明書が発行されていますが、有効期間はまだ許容されていません。

     - Revoked証明書が取り消されました。

- Expired期限切れの証明書の有効期間が終了しました。
- Revoked and Expired証明書は有効期間を渡し、取り消されました。

- Subject Name特定の所有者に属する証明書を一覧表示します。このフィールドでワイルドカードを使用できます。

注記

Certificate System 証明書要求フォームは、共通名、組織単位、および要求側の name フィールドの UTF-8 文字をすべてサポートします。共通名および組織単位フィールドは、証明書のサブジェクト名に含まれています。これは、サブジェクト名の検索で UTF-8 文字がサポートされていることを意味します。

このサポートには、国際化されたドメイン名のサポートは含まれません。

- Revocation Information特定の期間、特定のエージェント、または特定の理由で取り消された証明書を一覧表示します。たとえば、エージェントは、2005 年 7 月から 4 月の間に取り消されたすべての証明書、またはユーザー名 **admin** を持つエージェントによって取り消されたすべての証明書を一覧表示できます。
  - 期間内に取り消された証明書を一覧表示するには、ドロップダウンリストから日、月、および年を選択して、期間の開始と終了を識別します。
  - 特定のエージェントによって取り消された証明書を一覧表示するには、エージェントの名前を入力します。このフィールドではワイルドカードを使用できます。
  - 特定の理由で失効した証明書を一覧表示するには、リストから失効理由を選択します。
- Issuing Information特定の期間または特定のエージェントによって発行された証明書を一覧表示します。たとえば、エージェントは、2005 年 7 月から 4 月の間に発行されたすべての証明書、またはユーザー名 **jsmith** を使用してエージェントによって発行されたすべての証明書を一覧表示できます。
  - 期間内に発行された証明書を一覧表示するには、ドロップダウンリストから日、月、および年を選択して、期間の開始と終了を識別します。
  - 特定のエージェントによって発行された証明書を一覧表示するには、エージェントの名前を入力します。このフィールドではワイルドカードを使用できます。
  - 特定のプロファイルで登録された証明書を一覧表示するには、プロファイルの名前を入力します。
- Dates of Validity特定の期間に有効または有効期限が切れる証明書を一覧表示します。たとえば、エージェントは、2003 年 6 月 1 日に有効になったすべての証明書、または 2006 年 1 月 1 日から 2006 年 6 月 1 日の間に期限切れになったすべての証明書を一覧表示できます。

  有効期間が 1 か月未満のすべての証明書など、一定期間の有効期間を持つ証明書を一覧表示することもできます。
  - 一定期間内に有効または期限切れになる証明書を一覧表示するには、ドロップダウンリストから日、月、および年を選択して、期間の開始と終了を識別します。

- 特定の期間の有効期間を持つ証明書を一覧表示するには、ドロップダウンリストから **Not greater than** または **Not less than** を選択し、数字を入力して、ドロップダウンリストから時間単位を選択します（日、週、月、または年）。

- **基本の制約。**は、Basic Constraints 拡張機能に基づく CA 証明書を示しています。

- **Type**下位 CA のすべての証明書など、特定の種類の証明書を一覧表示します。この検索は、タイプ情報を格納する Netscape Certificate Type 拡張子を含む証明書に対してのみ機能します。それぞれのタイプについて、ドロップダウンリストから選択して、そのタイプが **On**、**Off**、または **Do Not Care** である証明書を検索します。

4. 特定のサブジェクト名を持つ証明書を検索するには、サブジェクト名 セクション **を使用** します。チェックボックスを選択し、サブジェクト名の基準を入力します。含まれている検索基準の値を入力し、その他は空白のままにします。

   標準のタグまたはコンポーネントは以下のとおりです。

   - **メールアドレス。**メールアドレスで検索を絞り込みます。

   - **Common name**特定の個人またはサーバーに関連付けられた証明書を見つけます。

   - **UserID**証明書の所有者のユーザー ID で証明書を検索します。

   - **Organization unit**組織内の特定の部門、部門、またはユニットに検索を絞り込みます。

   - **Organization**組織による検索を制限します。

   - **Locality**市区町村などの地域で絞り込みます。

   - **State**状態またはプロイエンス別に検索を絞り込みます。

   - **Country**国別に検索を絞り込みます。**US** などの 2 文字の国コードを使用します。

   注記

   Certificate System 証明書要求フォームは、共通名および組織単位フィールドですべての UTF-8 文字をサポートします。共通名および組織単位フィールドは、証明書のサブジェクト名に含まれています。これは、サブジェクト名またはサブジェクト名の要素の検索で UTF-8 文字がサポートされることを意味します。

   このサポートには、電子メールアドレスなどの国際化ドメイン名のサポートは含まれません。

5. サーバーが照合するフィールド値を入力したら、実行する検索のタイプを指定します。

   - 証明書サブジェクト名の正確な検索は、指定された正確なコンポーネントと一致し、空白のままになっているコンポーネントは含まれません。このタイプの検索では、ワイルドカードは使用できません。

   - 証明書のサブジェクト名の部分検索は、指定されたコンポーネントと一致しますが、返された証明書には、空白のままにされたコンポーネントの値が含まれている場合もあります。このタイプの検索では、任意の単一文字とアスタリスク(*)を照合し、任意の文字列の文字に一致させることで、このタイプの検索でワイルドカードパターンを使用できます。

注記

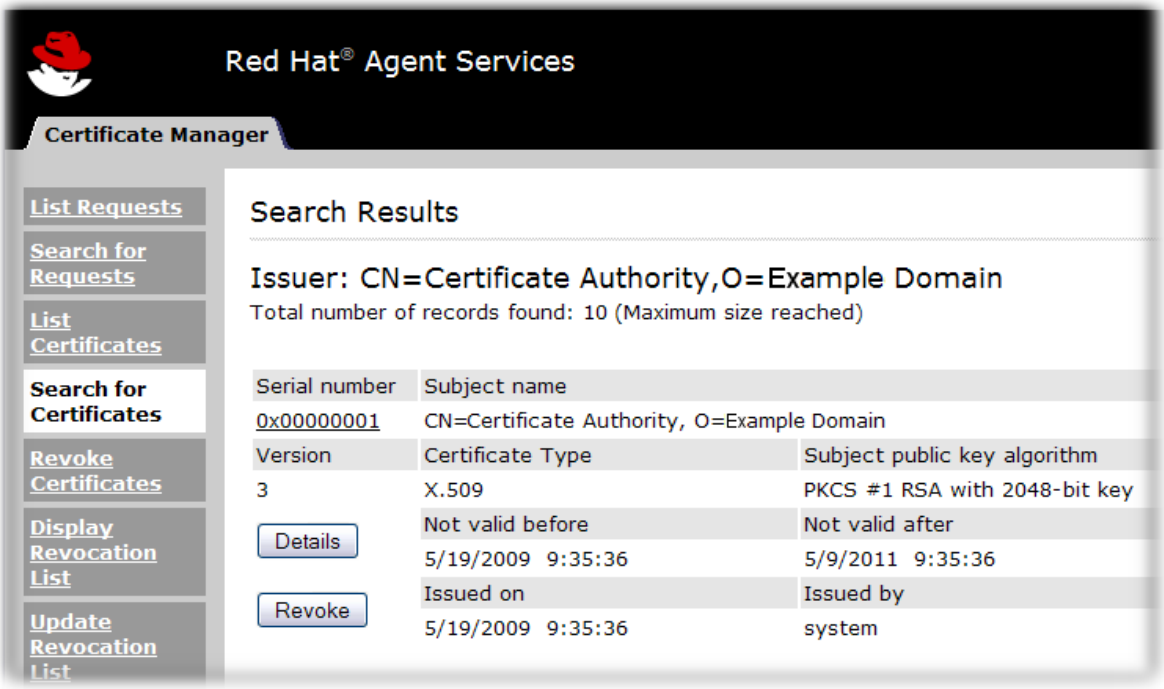
検索フィールドに単一のアスタリスクを配置することは、そのコンポーネントが証明書のサブジェクト名に含まれている必要があるが、任意の値を持つことができることを意味します。フィールドが存在するかどうかが問題でない場合は、フィールドを空白のままにします。

6. 検索条件を入力したら、フォームの一番下までスクロールし、指定された条件に一致する返される証明書の数を入力します。

   返される証明書の数を設定すると、その数までの検索条件に一致する最初に見つかった証明書が返されます。検索に時間制限を秒単位で指定することもできます。

7. **Find** をクリックします。

8. **Search Results** フォームが表示され、検索条件に一致する証明書の一覧が表示されます。リストで証明書を選択して、詳細を確認します。詳細は、「証明書の詳細の調査」 を参照してください。



6.2.2.3. 証明書の詳細の調査

1. エージェントサービスページで、**List Certificates** または **Search for Certificates** をクリックし、検索条件を指定して **Find** をクリックし、証明書の一覧を表示します。

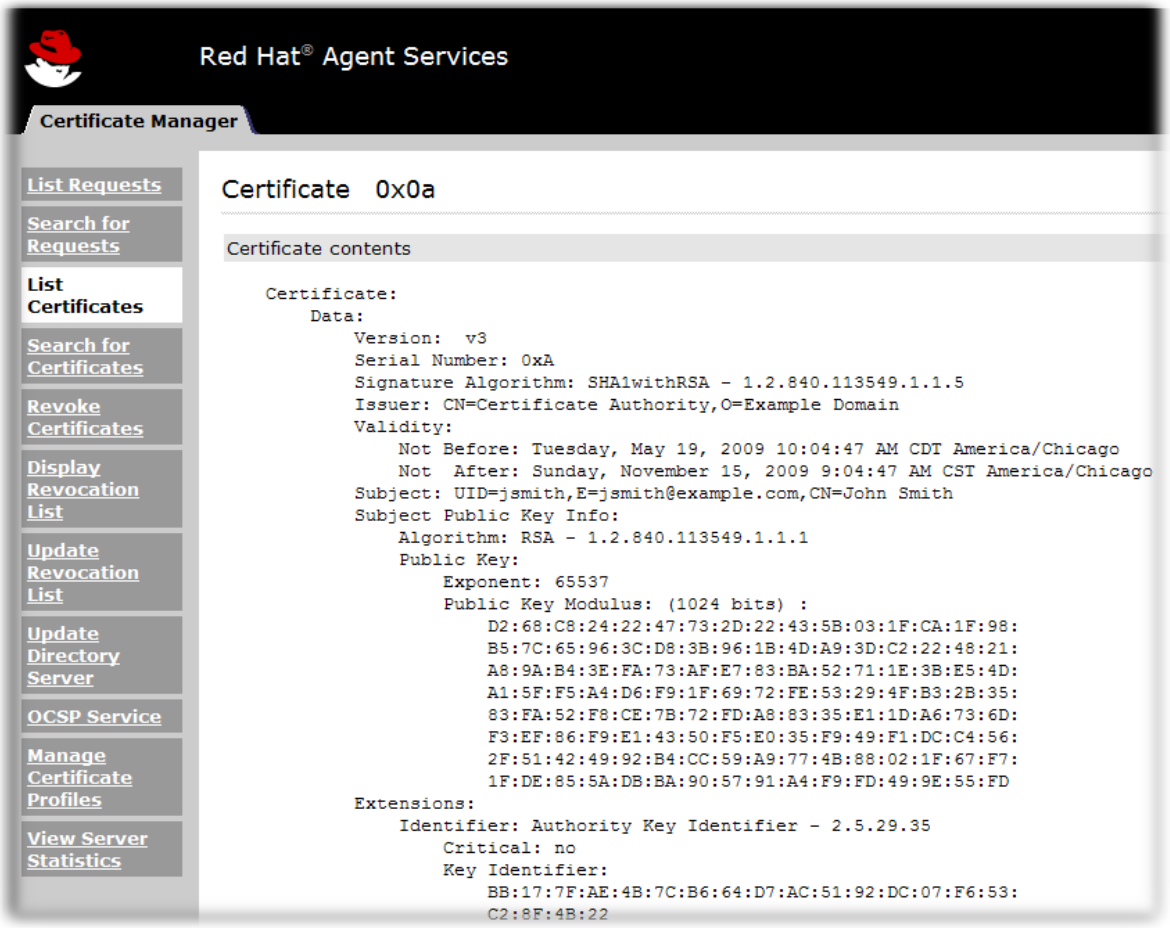
2. **Search Results** フォームで、検査する証明書を選択します。

   目的の証明書が表示されない場合は、リストの下部までスクロールし、返される証明書の数を追加で指定し、**Find** をクリックします。システムは、元の検索条件に一致する次の数まで、次の証明書を表示します。

3. 証明書を選択したら、エントリーの左側にある **Details** ボタンをクリックします。

4. **Certificate** ページには、選択した証明書の詳細な内容と、サーバーまたは Web ブラウザーに証明書をインストールする手順が表示されます。

図6.2 証明書の詳細



5. 証明書は、**Certificate** ページの下部の **Installing this certificate in a server** の見出しの下に、base-64 でエンコードされた形式で表示されます。

6.2.2.4. 証明書の取り消し

Certificate Manager エージェントのみが、独自の証明書以外の証明書を取り消すことができます。以下のいずれかの状況が発生した場合は、証明書を取り消す必要があります。

- 証明書の所有者はステータスを変更し、証明書を使用する権限がなくなりました。
- 証明書の所有者の秘密鍵が侵害されました。

証明書を取り消す必要があるのは、これら 2 つの理由だけではありません。証明書を取り消すには、6 つの理由があります。

1 つ以上の証明書を取り消すには、**Revoke Certificates** ボタンを使用して取り消す証明書を検索します。検索は **Search for Certificates** フォームを介して検索に似ていますが、この検索によって返される **Search Results** フォームは、返される証明書のいずれかまたはすべてを取り消すオプションを提供します。

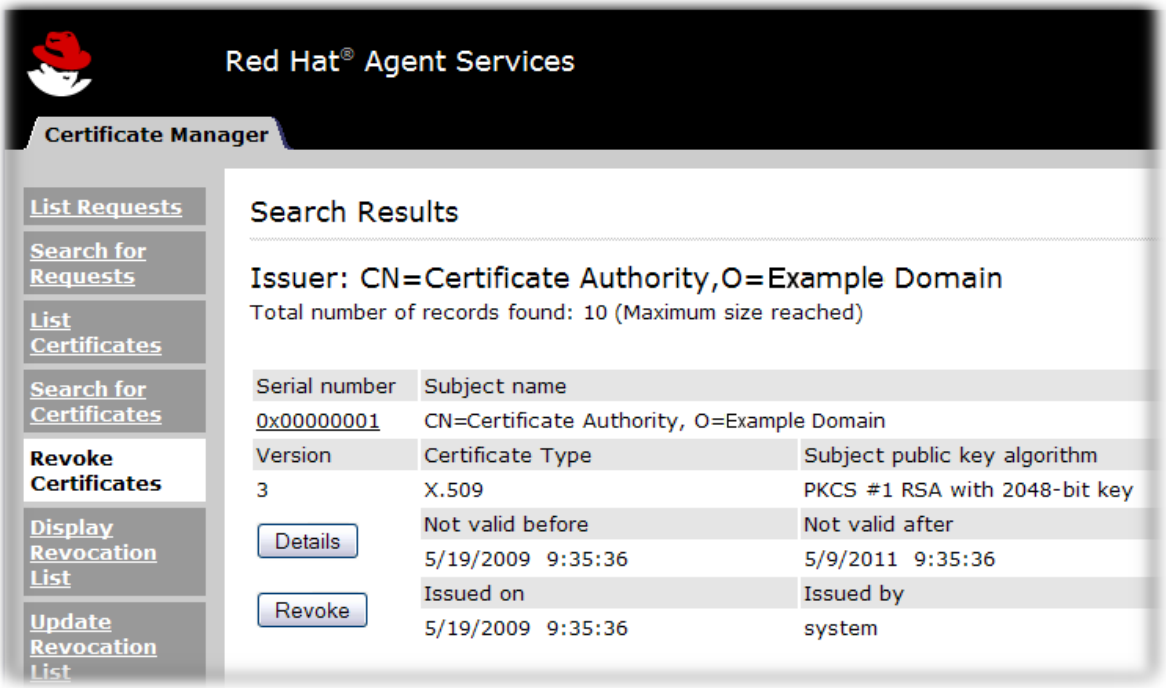
6.2.2.4.1. 証明書の取り消し

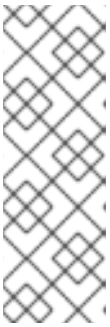
1. Certificate Manager エージェントサービスページを開きます。
2. **Revoke Certificates** をクリックします。

注記

表示される検索フォームには、**Search for Certificates** フォームと同じ検索条件セクションがあります。

3. セクションのチェックボックスを選択して検索条件を指定し、必要な情報を入力します。
4. フォームの下部までスクロールし、一致する証明書の数が表示されるように設定します。
5. **Find** をクリックします。
6. 検索は一致する証明書の一覧を返します。一覧内の 1 つまたはすべての証明書を取り消すことができます。





ヒント

検索条件が非常に固有で、返されるすべての証明書が取り消される場合は、ページの下部にある **Revoke ALL # Certificates** ボタンをクリックします。ボタンに表示される数は、検索によって返される証明書の合計数です。通常、これは現在のページに表示される証明書の数よりも大きくなります。

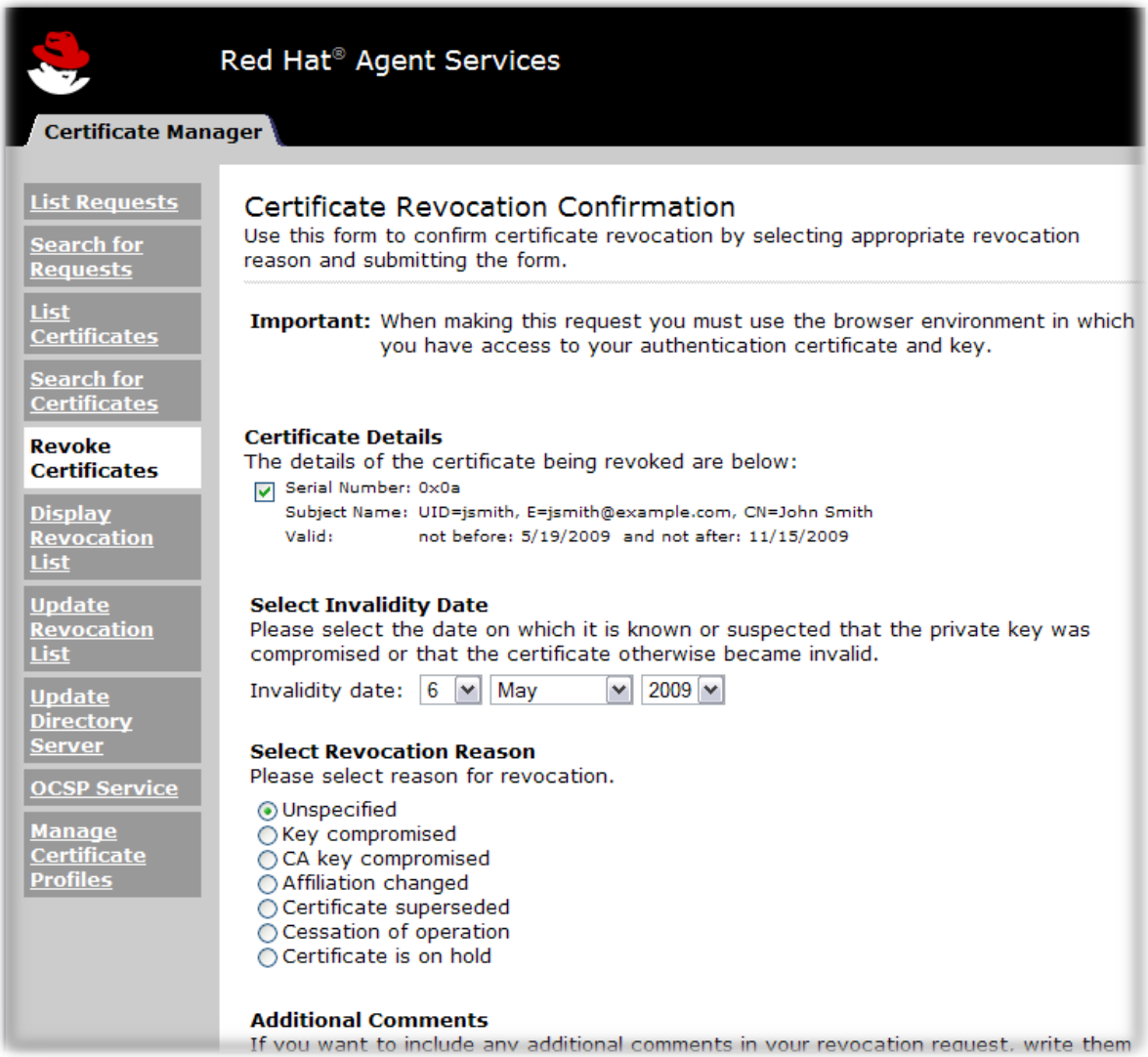
現在のページに表示されている証明書だけで **なく**、検索によって返されたすべての証明書を取り消す必要があることを確認します。

7. 取り消す証明書の横にある **Revoke** ボタンをクリックします。

注意

1つの証明書を呼び出すか、証明書の一覧を呼び出す場合でも、正しい証明書が選択されているか、リストに取り消される証明書のみが含まれていることに注意してください。失効操作が確認されたら、元に戻す方法はありません。

8. 無効な日付を選択します。無効な日付は、ユーザーの秘密鍵が侵害された、または証明書が無効になったことを疑われる日付です。ドロップダウンリストのセットにより、エージェントが正しい無効な日付を選択できます。



9. 失効の理由を選択します。

- 侵害された鍵
- CA キーが侵害されました
- 所属が変更
- 証明書が置き換えられました

- 運用停止
- 証明書は保持中です

10. 追加のコメントを入力します。コメントは失効リクエストに含まれます。

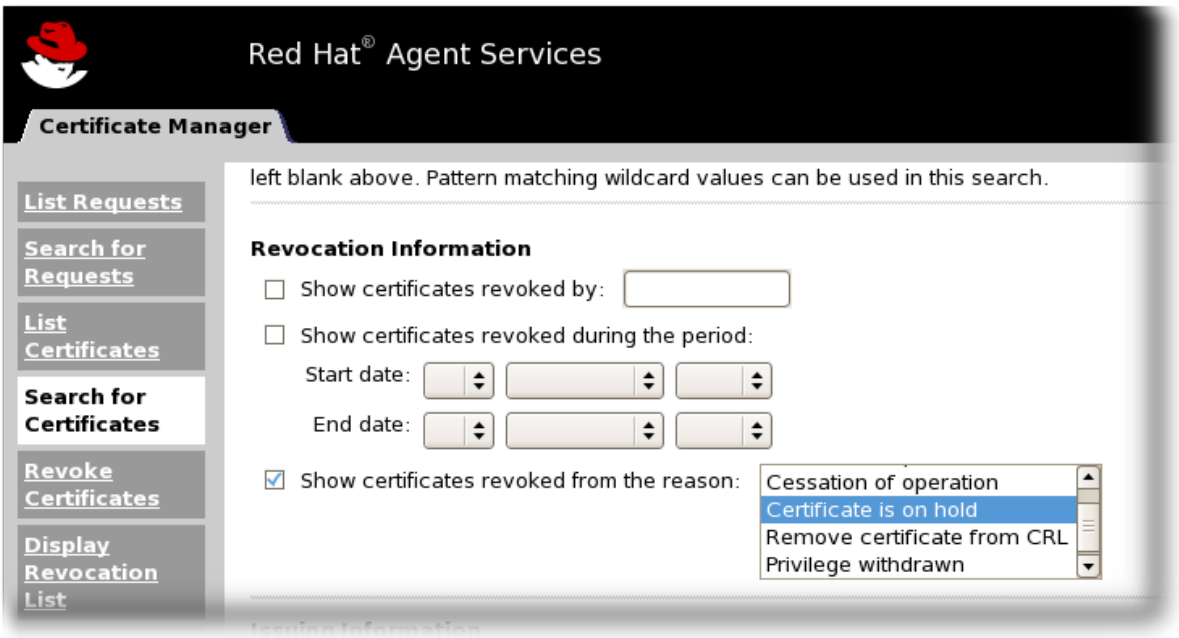
失効要求が送信されると、それは自動的に承認され、証明書は失効します。失効要求は、ステータスが **Completed** のリクエストを一覧表示して表示されます。

6.2.2.4.2. 証明書の保留解除

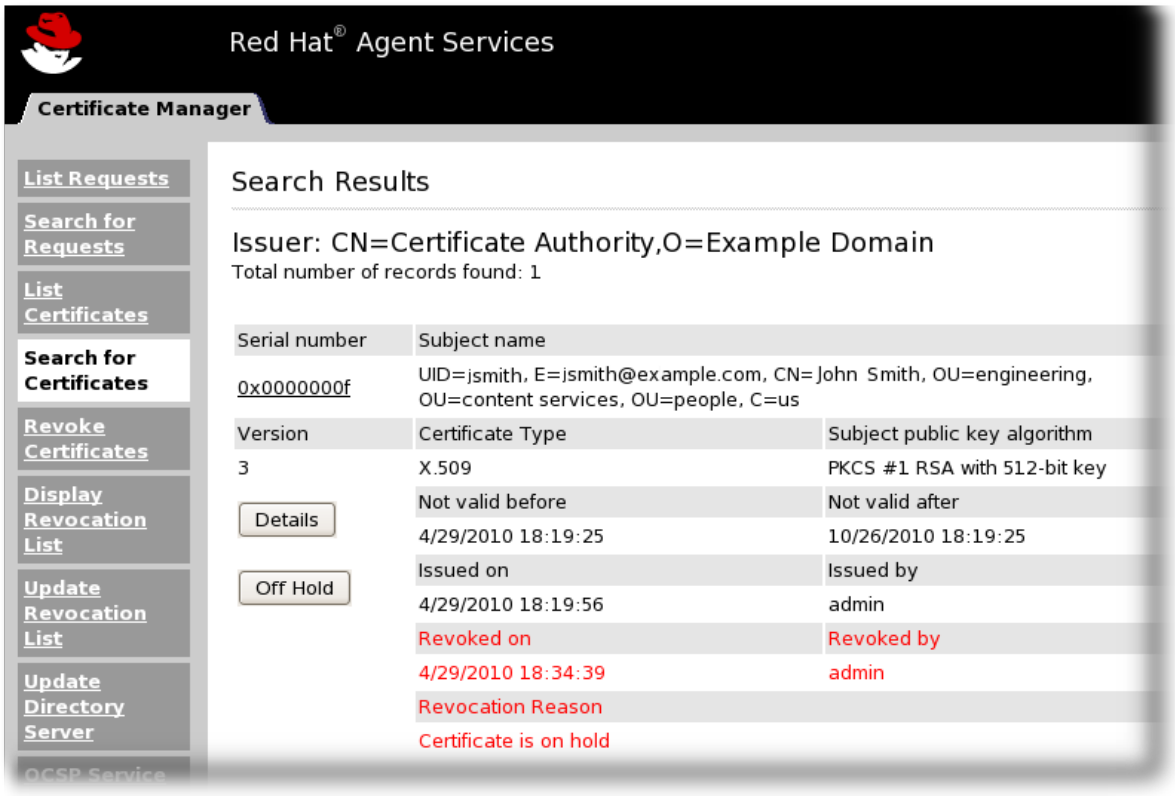
証明書にアクセスできないため、失効したものとして扱う必要がある場合がありますが、その証明書は回復できます。たとえば、あるユーザーがフラッシュドライブに保存されている個人の電子メール証明書を持っていて、それを誤って家に置いてしまった場合があります。証明書は侵害されていませんが、一時的に停止する必要があります。

この証明書は、保持時に一時的に取り消すことができます (「証明書の取り消し」 にあるように、証明書を取り消す際に指定されるオプションの 1 つ)。忘れたフラッシュドライブが取り出されたときなど、後でその証明書の保留を解除して、再びアクティブにすることができます。

1. 「証明書の検索 (詳細)」 にあるように、on hold 証明書を検索します。**Revocation Information** セクションまでスクロールし、検索条件として **Certificate is on hold** revocation reason を設定します。



2. 結果一覧で、証明書の **Off Hold** ボタンをクリックして保持します。



6.2.2.5. 証明書失効リストの管理

証明書を取り消すと、証明書が有効でなくなったことを他のユーザーに通知します。この通知は、証明書失効リスト (CRL) と呼ばれる **certificate revocation list** を LDAP ディレクトリーまたはフラットファイルに公開して行います。このリストは公開されているため、失効した証明書が正しく使用されていないことを確認してください。

6.2.2.5.1. CRL の表示または検証

最新の CRL でディレクトリーを手動で更新する前に、CRL を表示または検証する必要がある場合があります。CRL を表示または表示するには、以下を行います。

1. Certificate Manager エージェントサービスページに移動します。
2. **Display Certificate Revocation List** をクリックして、CRL を表示するためのフォームを表示します。
3. 表示する CRL を選択します。管理者が複数の発行ポイントを作成した場合、それらは **Issuing point** ドロップダウンリストに一覧表示されます。それ以外の場合は、マスター CRL のみが表示されます。
4. **Display Type** メニューからオプションの 1 つを選択して、CRL の表示方法を選択します。このメニューの選択は以下のとおりです。
   - **Cached CRL**CRL 自体からではなく、キャッシュから CRL を表示します。このオプションは、CRL 全体を表示するよりも速く結果を表示します。
   - **Entire CRL**CRL 全体を取得して表示します。
   - **CRL header**CRL ヘッダーのみを取得して表示します。
   - **Base 64 Encoded**base-64 でエンコードされた形式で CRL を取得し、表示します。

- **Delta CRL**デルタ CRL を取得して表示します。デルタ CRL は、最後の CRL が公開されてから新しい失効のみを示す CRL のサブセットです。このオプションは、デルタ CRL の生成が有効になっている場合にのみ利用できます。

5. 選択した CRL を確認するには、**Display** をクリックします。

   CRL がブラウザーウィンドウに表示されます。これにより、エージェントは特定の証明書が (シリアル番号で) リストに表示されているかどうかを確認し、最後の更新以降に取り消された証明書の総数、最後の更新以降に保留が解除された証明書の総数などの最近の変更を確認できます。、および最後の更新以降に有効期限が切れた証明書の総数。

6.2.2.5.2. CRL の更新

CRL の自動生成のスケジュールが有効になっている場合は、CRL を自動的に更新できます。スケジュールでは、設定された時間スケジュールで、または証明書の失効があるたびに CRL が生成されるように設定できます。

同様に、CRL 発行が有効になっている場合は、CRL も自動的に発行できます。

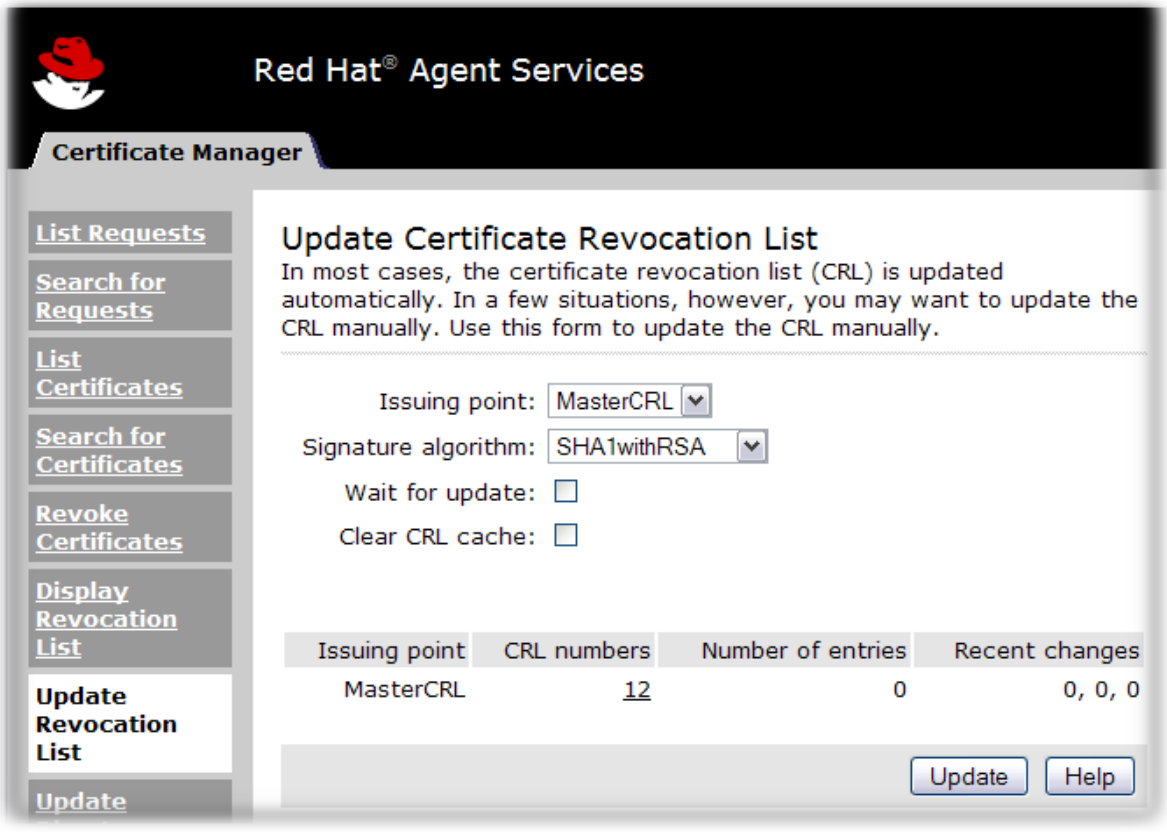
場合によっては、システムがダウンした後にリストを更新したり、期限切れの証明書を削除してファイルサイズを縮小したりするなど、CRL を手動で更新する必要があります。(期限切れの証明書は、有効期限のために既に無効になっているため、CRL に含める必要はありません。) Certificate Manager エージェントのみが CRL を手動で更新できます。

CRL を手動で更新するには:

1. Certificate Manager エージェントサービスページを開きます。

2. **Update Revocation List** をクリックし、CRL を更新するためのフォームを表示します。

   図6.3 証明書失効リストの更新



3. CRL を更新する CRL 発行ポイントを選択します。1 つの CA に対して複数の発行ポイントを設定できます。

4. 新しい CRL の署名に使用するアルゴリズムを選択します。アルゴリズムを選択する前に、この CRL の読み取りまたは表示が必要なシステムまたはネットワークアプリケーションがそのアルゴリズムをサポートしていることを確認してください。

   - RSA (SHA-256)
   - RSA (SHA-384)
   - RSA (SHA 512)

   アルゴリズムを選択する前に、Certificate System でそのアルゴリズムが有効になっていることを確認してください。Certificate System 管理者には、その情報があります。

5. **Update** をクリックして、最新の証明書失効情報で CRL を更新します。

6.2.3. Web UI を使用したユーザーとしての Own 証明書での失効の実行

証明書を取り消すと、有効期限が切れる前に証明書が無効になります。これは、証明書が失われたり、侵害されたり、不要になった場合に必要になることがあります。

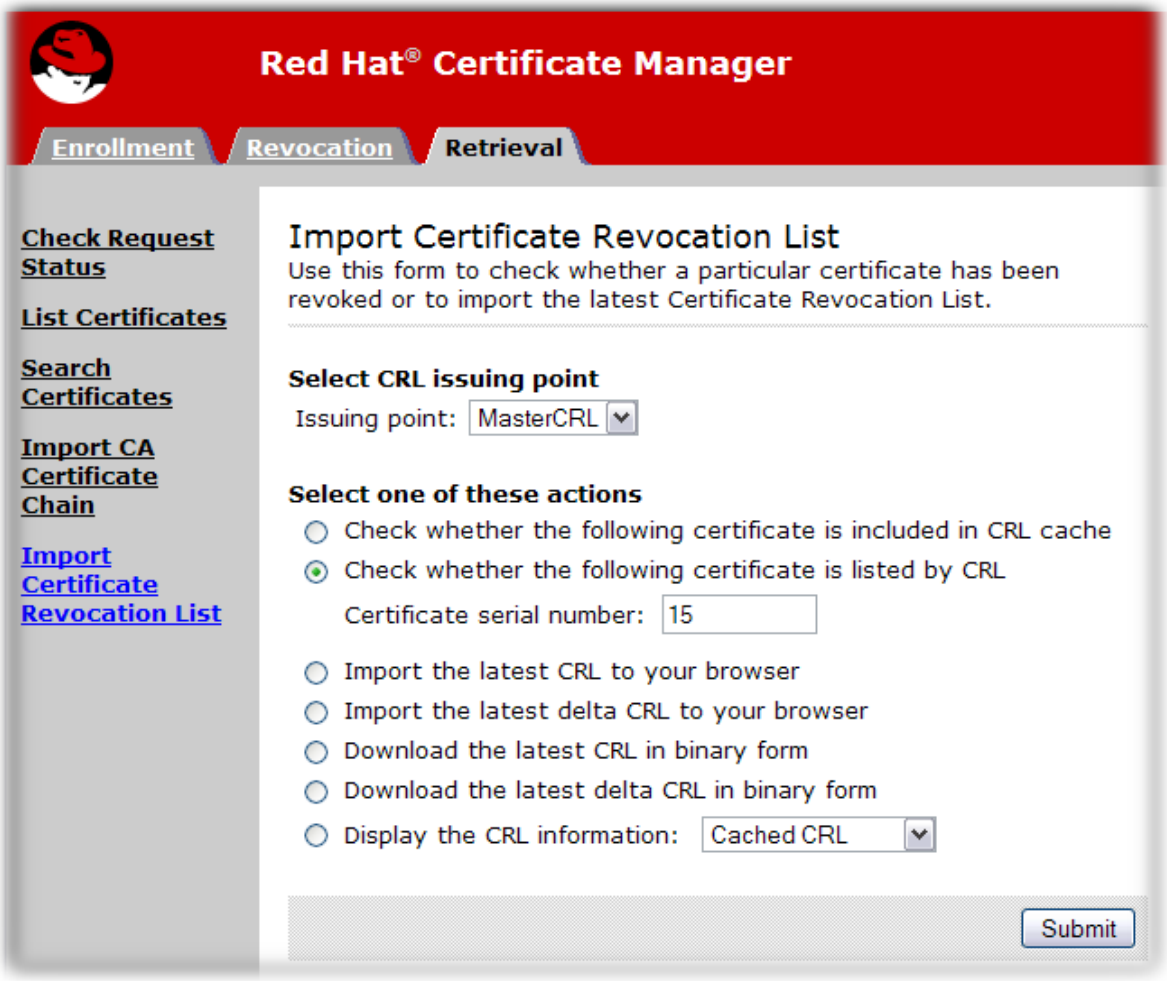
6.2.3.1. ユーザー証明書の取り消し

1. **Revocation** タブをクリックします。
2. **User Certificate** リンクをクリックします。
3. 証明書が取り消される理由を選択し、**Submit** をクリックします。

4. 一覧から取り消す証明書を選択します。

6.2.3.2. 証明書が取り消されているかどうかの確認

1. **Retrieval** タブをクリックします。
2. **Import Certificate Revocation List** リンクをクリックします。
3. **Check whether the following certificate is included in CRL cache** または **Check whether the following certificate is listed by CRL** のラジオボタンを選択し、証明書のシリアル番号を入力します。



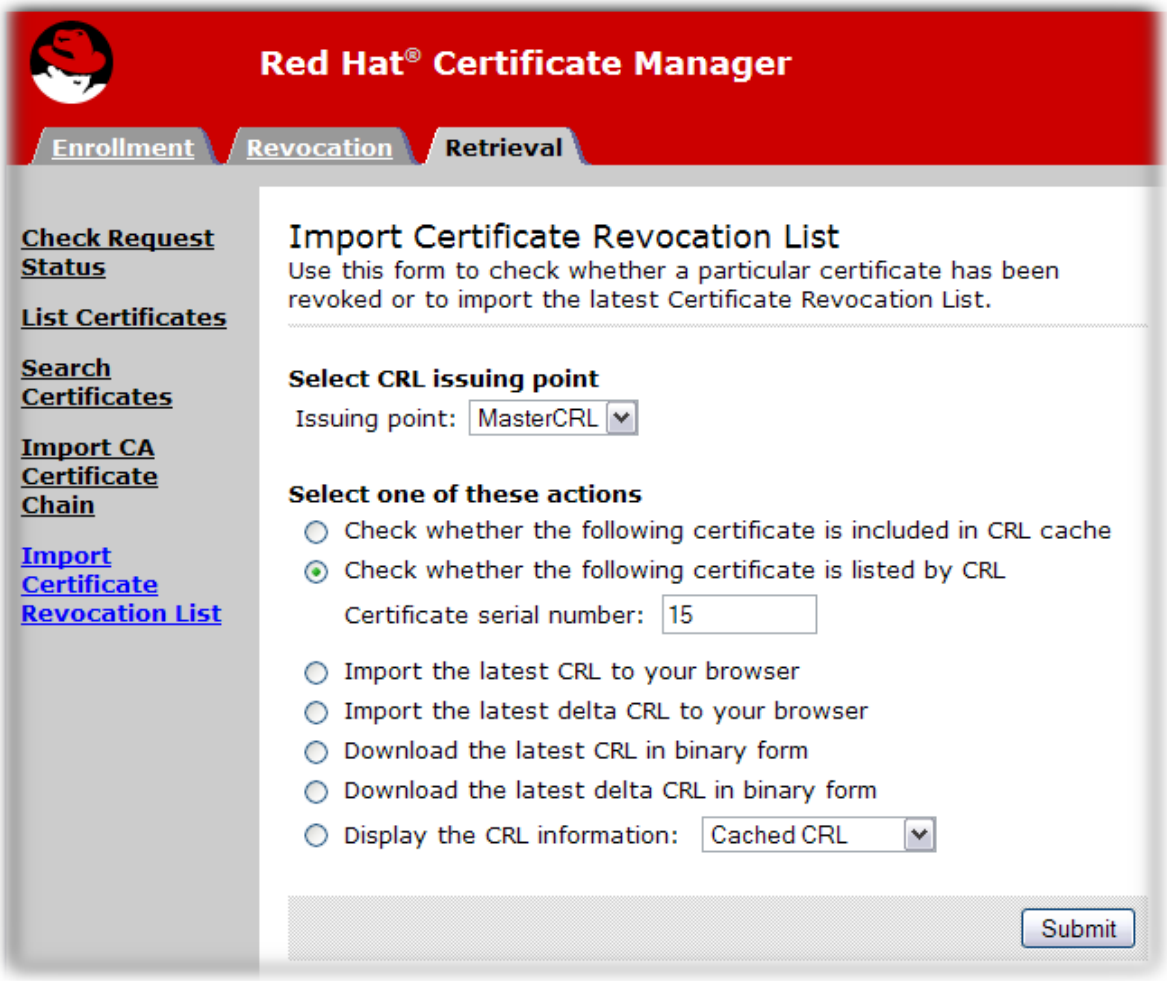
4. **送信** ボタンをクリックします。

   メッセージは、証明書が CRL にリストされていないことを示すか、証明書が含まれる CRL の情報を提供します。

6.2.3.3. CRL のダウンロードおよびインポート

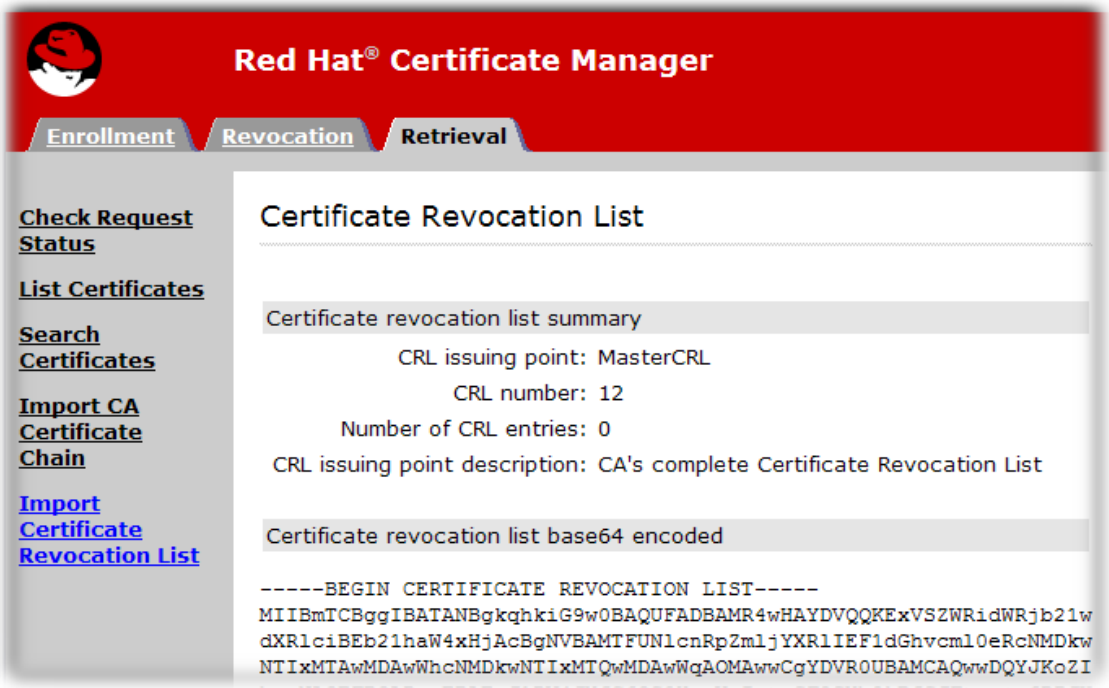
証明書失効リスト (CRL) をダウンロードして、Web クライアント、アプリケーション、またはマシンにインストールできます。それらは表示して、取り消された証明書を確認することもできます。

1. **Retrieval** タブをクリックします。
2. **Import Certificate Revocation List** リンクをクリックします。
3. ラジオボタンを選択して、CRL を表示、ダウンロード、またはインポートします。



- CRL をブラウザーにインポートするか、ダウンロードして保存するには、適切なラジオボタンを選択します。完全な CRL またはデルタ CRL をダウンロード/インポートする方法は 2 つあります。デルタ CRL は、最後に CRL が生成されてから取り消された証明書の一覧のみをインポート/ダウンロードします。

- CRL を表示するには、**CRL 情報の表示** を選択し、表示する CRL サブセット(**発行ポイント**と呼ばれる)を選択します。これは、含まれる証明書の数を含む CRL 情報を示しています。



4. **送信** ボタンをクリックします。

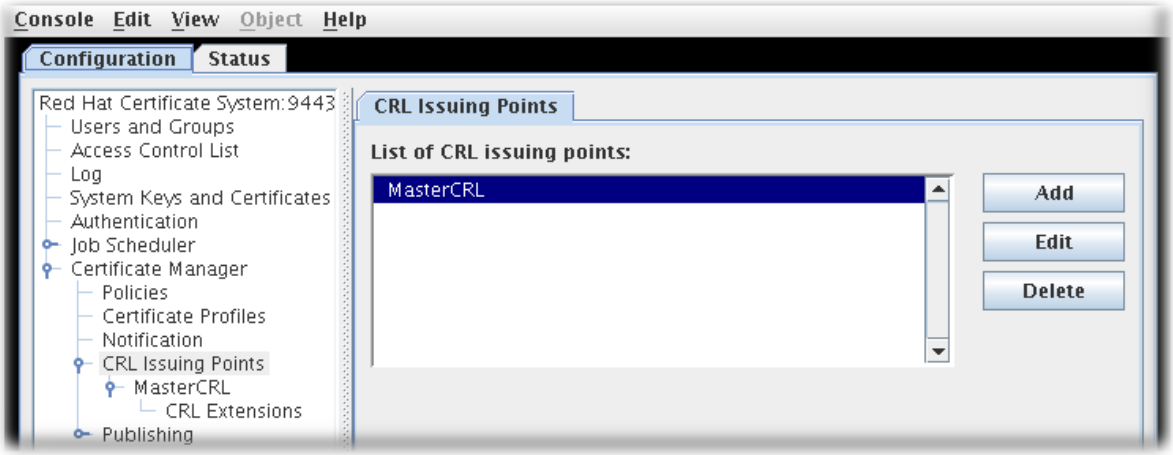
5. ファイルを保存するか、インポート操作を承認します。

6.3. CRL の実行

1. Certificate Manager は、その OCSP 署名キーを使用して CRL に署名します。CRL に個別の署名キーペアを使用するには、CRL 署名キーを設定し、このキーを使用して CRL に署名するように Certificate Manager の設定を変更します。詳細は、Red Hat Certificate System の計画、インストール、およびデプロイメントのガイド 9.2.3.11「別の証明書を使用して CRL に署名するように CA を設定する」を参照してください。

2. CRL 発行ポイントの設定発行ポイントは、マスター CRL に対してすでにセットアップされ、有効にされています。

図6.4 デフォルトの CRL 発行ポイント



CRL の追加の発行ポイントを作成できます。詳細は、「発行ポイントの設定」を参照してください。

発行ポイントを設定して CRL のリストを定義するときに設定したオプションに応じて、発行ポイントが作成できる CRL には 5 つのタイプがあります。

- **マスター CRL** には、CA 全体から失効した証明書の一覧が含まれます。
- **ARL** は、失効した CA 証明書のみが含まれる Authority Revocation List です。
- **期限切れの証明書を持つ CRL** には、CRL で有効期限が切れた証明書が含まれます。
- **証明書プロファイルの CRL** は、最初に証明書を作成するために使用されるプロファイルに基づいて、失効した証明書を判別します。
- **理由コードによる CRL** は、失効した理由コードに基づいて、失効した証明書を判別します。

3. 各発行ポイントに CRL を設定します。詳細は、「各発行ポイントの CRL の設定」を参照してください。
4. 発行ポイントに設定された CRL 拡張機能を設定します。詳細は、「CRL 拡張機能の設定」を参照してください。
5. 発行ポイントの拡張を有効にすることにより、発行ポイントにデルタ CRL を設定するか、または発行ポイント **DeltaCRLIndicator** または **CRLNumber** の拡張を有効にします。
6. 発行先に関する情報が含まれるように **CRLDistributionPoint** 拡張機能を設定します。
7. ファイル、LDAP ディレクトリー、または OCSP レスポンダーへの公開 CRL を設定します。公開の設定の詳細は、*7章 証明書および CRL の公開* を参照してください。

6.3.1. 発行ポイントの設定

発行ポイントは、新しい CRL に含まれる証明書を定義します。マスター CRL 発行ポイントは、Certificate Manager の失効した証明書の一覧を含むマスター CRL 用にデフォルトで作成されます。

新規の発行ポイントを作成するには、以下の手順を実施します。

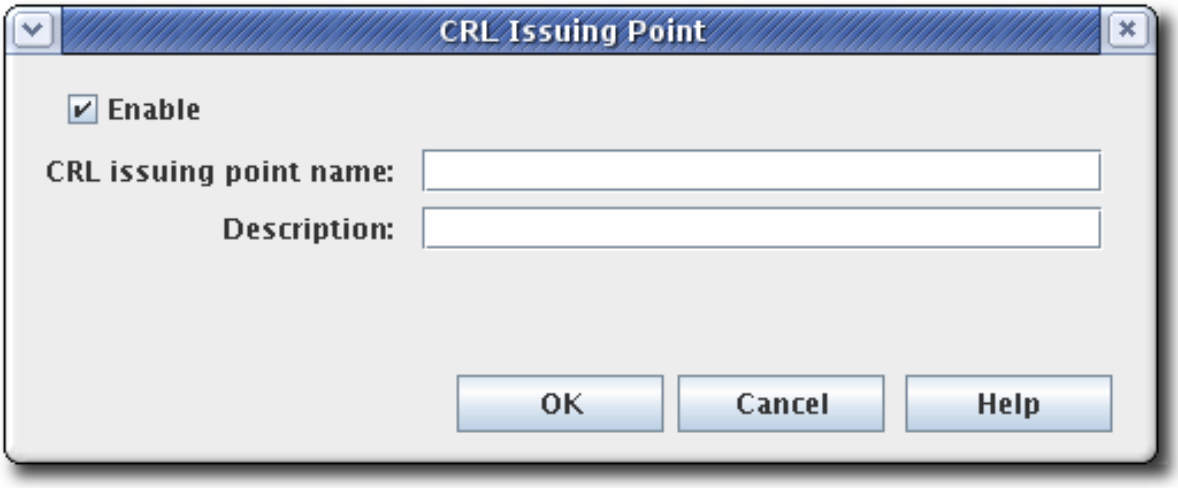
1. 証明書システムコンソールの起動

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

2. **Configuration** タブで、左側のナビゲーションメニューから **Certificate Manager** を展開します。次に、**CRL Issuing Points** を選択します。
3. 発行ポイントを編集するには、発行ポイントを選択して、**Edit** をクリックします。編集できるパラメーターは、発行ポイントの名前と、発行ポイントが有効か無効かだけです。

   発行ポイントを追加するには、**Add** をクリックします。CRL Issuing Point エディターウインドウが開きます。

図6.5 CRL Issuing Point エディター



注記

一部のフィールドがコンテンツを読み取るのに十分な大きさで表示されない場合は、コーナーの1つをドラッグしてウィンドウを拡大します。

以下のフィールドに入力します。

- **Enable**。選択した場合は発行ポイントを有効にします。無効にする場合は選択を解除します。
- **CRL Issuing Point name**。発行ポイントの名前を指定します。スペースは使用できません。
- **Description**。発行ポイントを説明します。

4. **OK** をクリックします。

新しい発行ポイントを表示して設定するには、CA コンソールを閉じ、その後にコンソールを再度開きます。新しい発行ポイントは、ナビゲーションツリーの **CRL Issuing Points** エントリーの下に一覧表示されます。

新しい発行ポイントに CRL を設定し、CRL と使用する CRL 拡張機能を設定します。発行ポイントの設定に関する詳細は、「各発行ポイントの CRL の設定」 を参照してください。CRL 拡張機能の設定に関する詳細は、「CRL 拡張機能の設定」 を参照してください。作成された CRL はすべて、エージェントサービスページの **Update Revocation List** ページに表示されます。

6.3.2. 各発行ポイントの CRL の設定

生成間隔、CRL バージョン、CRL 拡張、署名アルゴリズムなどの情報はすべて、発行ポイントの CRL 用に設定できます。CRL は発行ポイントごとに設定する必要があります。

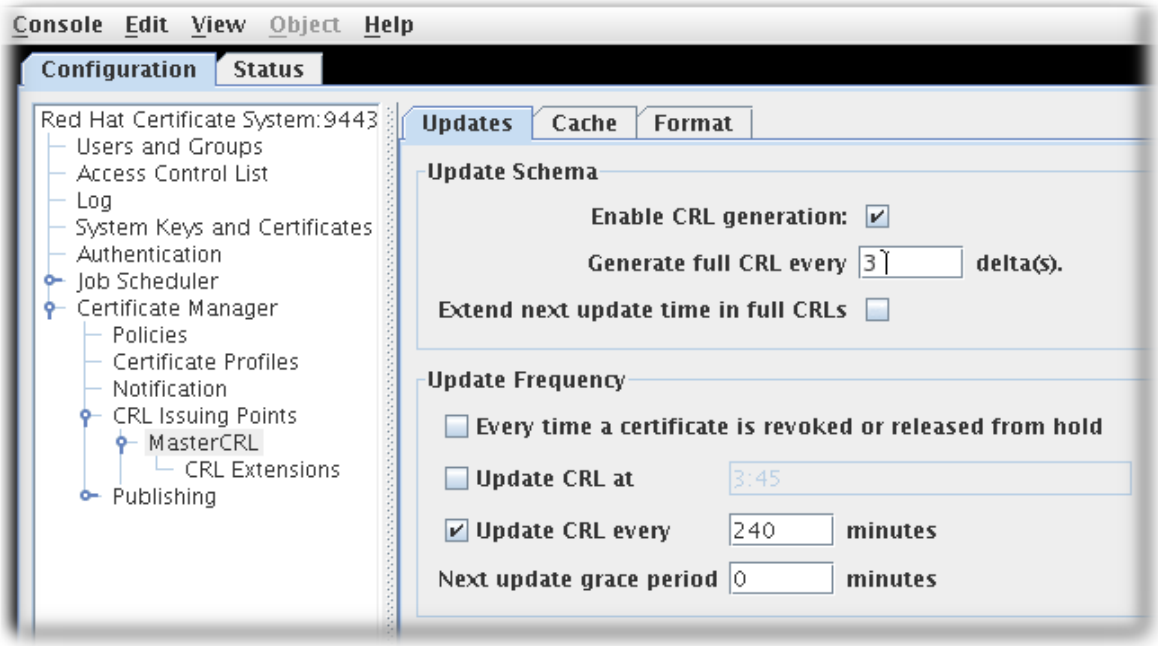
1. CA コンソールを開きます。

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

2. ナビゲーションツリーで、**Certificate Manager** を選択し、**CRL Issuing Points** を選択します。

3. **Issuing Points** エントリーの下に、発行ポイント名を選択します。

4. 発行ポイントの **Update** タブに情報を指定して、CRL の更新方法および頻度を設定します。このタブには、**Update Schema** および **Update Frequency** の 2 つのセクションがあります。



- **Update Schema** セクションには以下のオプションが含まれます。

  - **CRL 生成を有効にします**。このチェックボックスは、発行ポイントに CRL が生成されるかどうかを設定します。

  - **Generate full CRL every # delta(s)**。このフィールドは、変更の数に関連して CRL が作成された頻度を設定します。

  - **Extend next update time in full CRLs**。これにより、生成された CRL に **nextUpdate** フィールドを設定するオプションが提供されます。***nextUpdate*** パラメーターは、フル CRL かデルタ CRL かに関係なく、次の CRL が発行される日付を示します。フル CRL とデルタ CRL の組み合わせを使用している場合は、***Extend next update time in full CRLs*** を有効にすると、フル CRL の ***nextUpdate*** パラメーターに次の ***フル*** CRL が発行されるタイミングを表示させることができます。それ以外の場合は、フル CRL の ***nextUpdate*** パラメーターは、そのデルタが次に発行される CR L になるため、次の ***デルタ*** CRL がいつ発行されるかを示します。

- **Update Frequency** セクションは、CRL が生成され、ディレクトリーに発行されたときに異なる間隔を設定します。

  - **Every time a certificate is revoked or released from hold**。これにより、証明書を取り消すたびに Certificate Manager が CRL を生成するよう設定されます。Certificate Manager は、CRL が生成されるたびに、設定されたディレクトリーに CRL を発行しようとします。CRL の生成は、CRL のサイズが大きい場合に消費できます。証明書が取り消されるたびに CRL を生成するように Certificate Manager を設定すると、サーバーがかなりの時間使用される可能性があります。この間、サーバーは受け取った変更でディレクトリーを更新できなくなります。

    この設定は、標準的なインストールには推奨されません。このオプションは、サーバーが CRL をフラットファイルに発行したかどうかのテストなど、すぐに失効をテストするために選択する必要があります。

- **Update the CRL at**。このフィールドは、CRL を更新する必要がある毎日の時間を設定します。複数回指定するには、**01:50,04:55,06:55** などのコンマ区切りリストを入力します。複数日のスケジュールを入力するには、コンマ区切りのリストを入力して同じ日の時間を設定し、セミコロンで区切ったリストを入力して異なる日の時間を識別します。たとえば、これは、サイクルの 1 日目の 午前 1:50、4:55、および 6:55、そして 2 日目の午前 2:00、5:00、および午後 5:00 に失効を設定します。

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

- **CRL をすべて更新** します。このチェックボックスでは、フィールドに設定された間隔で CRL を生成できます。たとえば、毎日 CRL を発行するには、チェックボックスを選択して、このフィールドに **1440** を入力します。

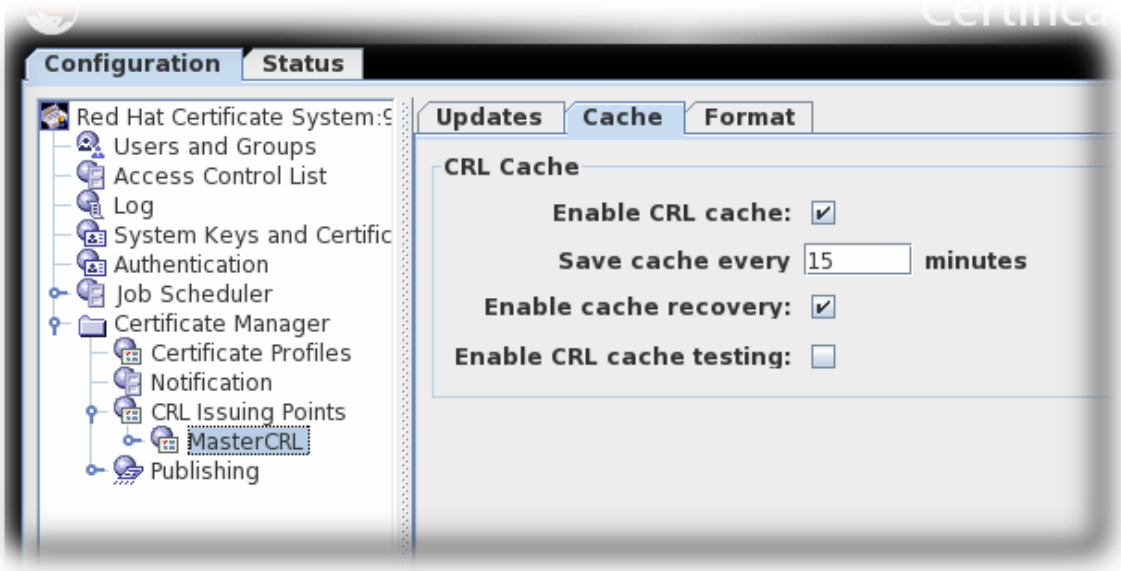
- **Next update grace period**。Certificate Manager が特定の頻度で CRL を更新する場合、サーバーは、CRL を作成して発行する時間を確保するために、次の更新時間までの猶予期間を持つように設定できます。たとえば、サーバーが 2 分の猶予期間で 20 分ごとに CRL を更新するように設定されていて、CRL が 16:00 に更新された場合、CRL は 16:18 に再更新されます。

重要

既知の問題により、現在フルおよびデルタの証明書失効リストのスケジュールを設定している場合、**Update CRL every time a certificate is revoked or released from hold** オプションでは、2 つの **grace period** 設定を記入する必要もあります。したがって、このオプションを選択するには、最初に **Update CRL every** オプションを選択して、する必要がありますし、**Next update grace period # minutes** ボックスに番号を入力する必要があります。

5. **Cache** タブは、キャッシュが有効であるかどうかとキャッシュ頻度を設定します。

図6.6 CRL キャッシュタブ



- **Enable CRL cache**。このチェックボックスは、デルタ CRL の作成に使用されるキャッシュを有効にします。キャッシュが無効になっている場合は、デルタ CRL は作成されません。キャッシュの詳細は、「証明書の失効について」 を参照してください。

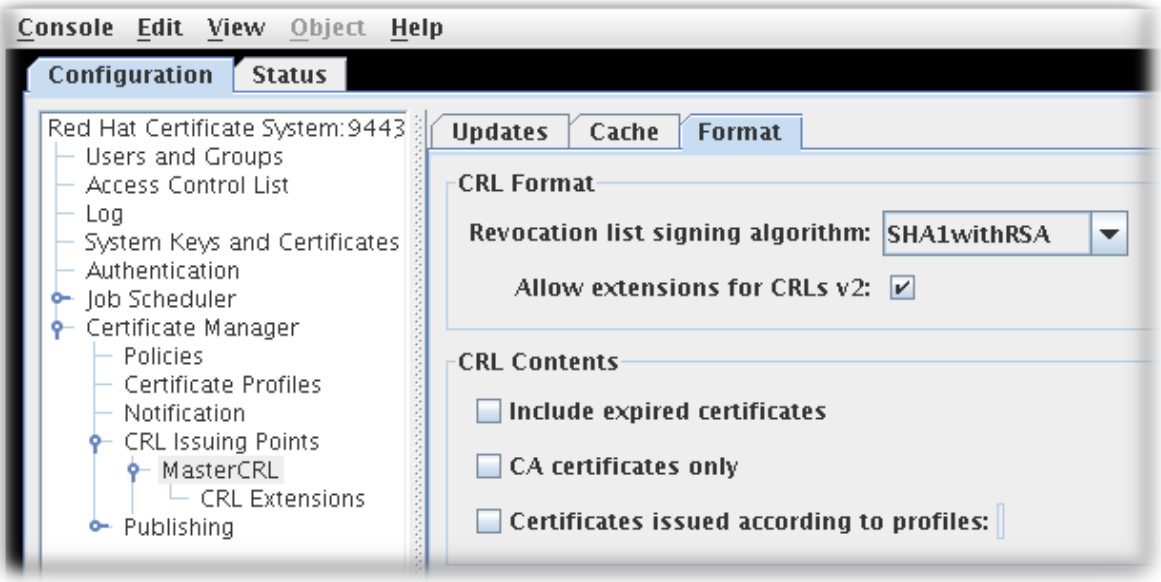
- **キャッシュを毎回更新** します。このフィールドは、キャッシュが内部データベースに書き込む頻度を設定します。証明書が取り消されるたびに、キャッシュをデータベースに書き出すには、**0** に設定します。

- **キャッシュリカバリーを有効** にします。このチェックボックスを選択すると、キャッシュを復元できます。

- **Enable CRL cache testing**。このチェックボックスは、特定の CRL 発行ポイントの CRL パフォーマンステストを有効にします。このオプションで生成された CRL は、デプロイした CA では使用しないでください。テスト目的で発行された CRL には、パフォーマンステストのみを目的として生成されたデータが含まれているためです。

6. **フォーマット** タブでは、作成される CRL のフォーマットおよびコンテンツを設定します。**CRL Format** および **CRL Contents** の 2 つのセクションがあります。

図6.7 CRL 形式タブ



- **CRL Format** セクションには、以下の 2 つのオプションがあります。

  - **Revocation list signing algorithm** は、CRL 暗号化を行うために許可された暗号のドロップダウンの一覧です。

  - **Allow extensions for CRL v2** するには、発行ポイントに CRL v2 拡張を有効にするチェックボックスがあります。これが有効な場合は、「CRL 拡張機能の設定」 で説明されている必要な CRL 拡張機能を設定します。

注記

CRL を作成するには、拡張機能を有効にする必要があります。

- **CRL Contents** セクションには、CRL に追加する証明書のタイプを設定する 3 つのチェックボックスがあります。

  - **期限切れの証明書** を含めます。これには、期限切れになった証明書が含まれます。これを有効にすると、失効した証明書に関する情報は、証明書の期限が切れた後も CRL に残ります。これが有効になっていないと、証明書の有効期限が切れると、失効した証明書に関する情報が削除されます。

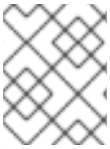
- **CA 証明書のみ**これには、CRL の CA 証明書のみが含まれます。このオプションを選択すると、失効した CA 証明書のみを一覧表示する Authority Revocation List (ARL) が作成されます。

- **プロファイルに従って発行された証明書**。これには、リストされたプロファイルに従って発行された証明書のみが含まれます。複数のプロファイルを指定するには、コンマ区切りのリストを入力します。

7. **Save** をクリックします。

8. この発行ポイントでは、拡張機能は可能で、設定できます。詳細は「CRL 拡張機能の設定」を参照してください。

6.3.3. CRL 拡張機能の設定



注記

拡張機能には、発行ポイントに CRLs v2 の **Allow extensions for CRLs v2** チェックボックスが選択されている場合にのみ、発行ポイントに必要です。

発行ポイントが作成されると、3 つの拡張機能 (**CRLReason**、**InvalidityDate**、および **CRLNumber**) が自動的に有効になります。その他の拡張は利用できますが、デフォルトで無効になっています。これは、有効化および変更できます。利用可能な CRL 拡張の詳細は、??? を参照してください。

CRL 拡張機能を設定するには、以下を行います。

1. CA コンソールを開きます。

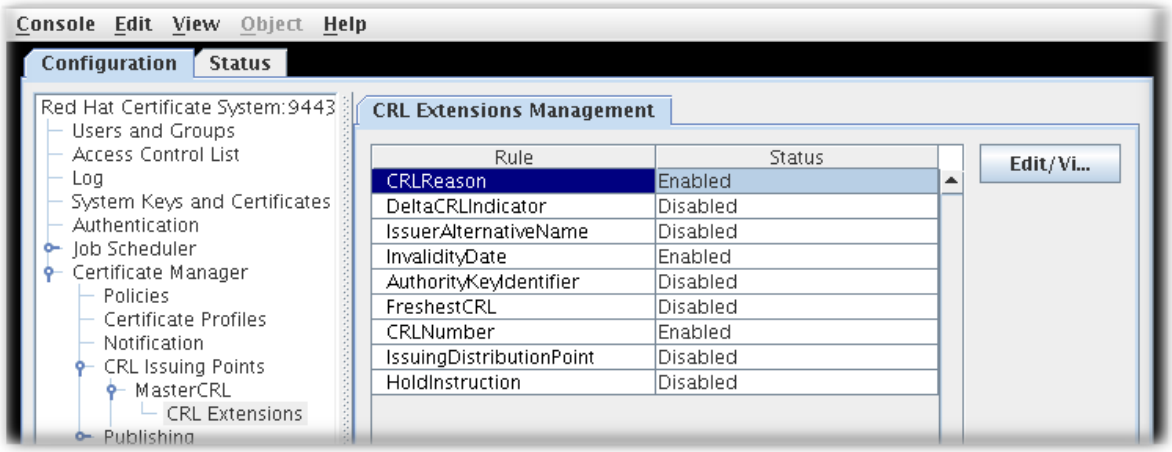
```
pkiconsole https://server.example.com:8443/ca
```

2. ナビゲーションツリーで、**Certificate Manager** を選択し、**CRL Issuing Points** を選択します。

3. **Issuing Points** エントリーの下にある発行ポイント名を選択し、発行ポイントの下にある **CRL 拡張** エントリーを選択します。

   右側のペインには、設定された拡張機能を一覧表示する **CRL Extensions Management** タブが表示されます。

図6.8 CRL 拡張機能



4. ルールを変更するには、ルールを選択し、**Edit/View** をクリックします。

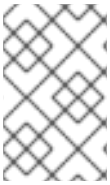
5. ほとんどの拡張には 2 つのオプションがあり、有効にして、重要なかどうかを設定します。詳細情報が必要なものもあります。必要な値をすべて指定します。各拡張機能およびそれらの拡張機能のパラメーターに関する詳細は、??? を参照してください。

6. **OK** をクリックします。

7. **Refresh** をクリックし、すべてのルールの更新されたステータスを表示します。

6.3.4. キャッシュからの CRL の生成

デフォルトでは、CRL は CA の内部データベースから生成されます。ただし、証明書が取り消されてメモリーに保持されるため、失効情報を収集できます。その後、この失効情報を使用して、メモリーから CRL を更新できます。内部データベースから CRL を生成するために必要なデータベース検索を省略すると、パフォーマンスが大幅に改善されます。



注記

キャッシュから CRL を生成する際のパフォーマンスの向上により、ほとんどの環境で ***enableCRLCache*** パラメーターが有効になります。ただし、実稼働環境ではこの ***Enable CRL cache testing*** パラメーターを有効に **しないでください**。

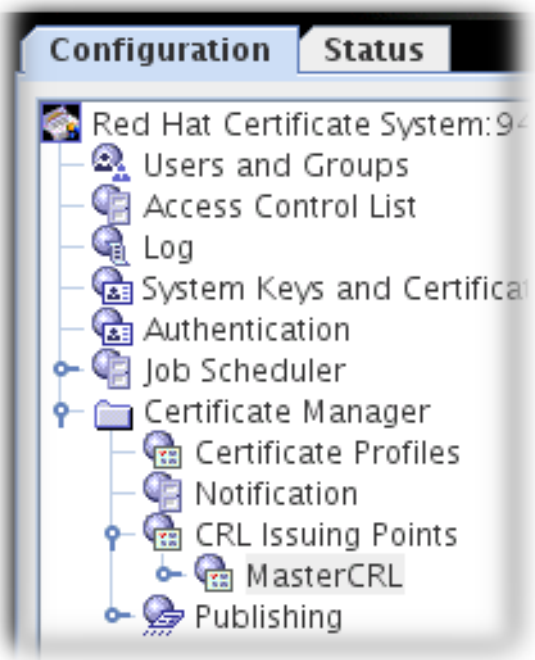
6.3.4.1. コンソールでのキャッシュからの CRL 生成の設定

1. コンソールを開きます。

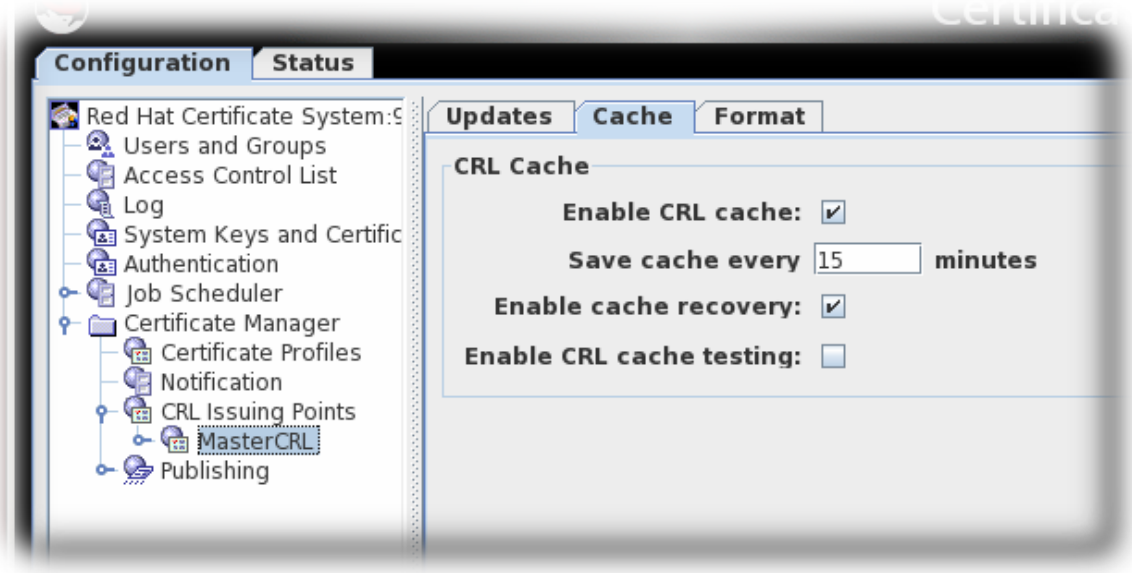
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、**Certificate Manager** フォルダーと **CRL Issuing Points** サブディレクトリーを展開します。

3. **MasterCRL** ノードを選択します。



4. **Enable CRL cache** を選択します。



5. 変更を保存します。

6.4. FULL および DELTA CRL スケジュールの設定

CRL は定期的に生成されます。「各発行ポイントの CRL の設定」 の設定でその期間は切り替わるものです。

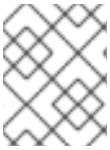
CRL は、時間ベースのスケジュールに従って発行されます。CRL は、証明書が失効するたびに、1 日の特定の時間帯に、または数十分に 1 回発行することができます。

時間ベースの CRL 生成スケジュールは、生成されるすべての CRL に適用されます。CRL には完全な CRL とデルタ CRL の 2 つの種類があります。完全な CRL には、取り消されたすべての証明書のレコードがありますが、デルタ CRL には、最後の CRL (デルタまたは完全) が生成されてから取り消された証明書のみが含まれます。

デフォルトでは、完全な CRL はスケジュールで指定した間隔で生成されます。正確な **delta** CRL を生成することで、完全な CRL を生成するまでに時間がかかる場合があります。生成間隔は **CRL スキーマ** で設定され、デルタと完全な CRL を生成するスキームを設定します。

たとえば、間隔が 3 に設定されている場合、生成される最初の CRL はフル CRL とデルタ CRL の両方になり、次の 2 つの世代の更新はデルタ CRL のみになり、4 番目の間隔は再びフル CRL とデルタ CRL の両方になります。つまり、3 番目の生成間隔はすべて完全な CRL とデルタ CRL の両方があります。

```
Interval   1, 2, 3, 4, 5, 6, 7 ...
Full CRL   1        4        7 ...
Delta CRL  1, 2, 3, 4, 5, 6, 7 ...
```



注記

完全な CRL に加えてデルタ CRL を生成するには、CRL キャッシュを有効にする必要があります。

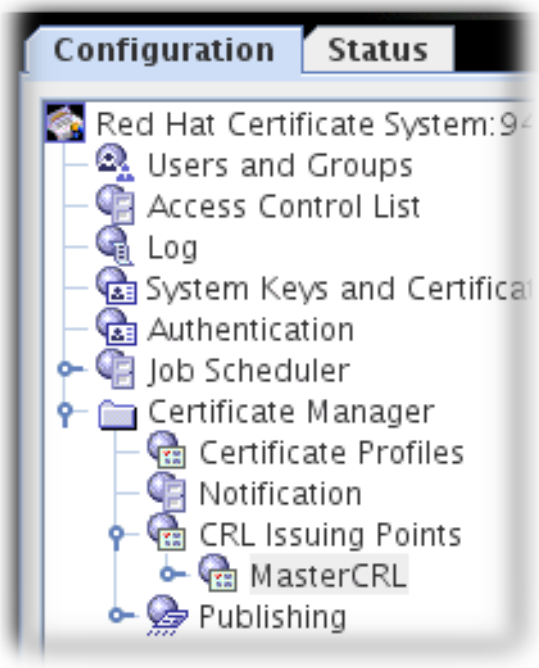
6.4.1. コンソールでの CRL 更新間隔の設定

1. コンソールを開きます。

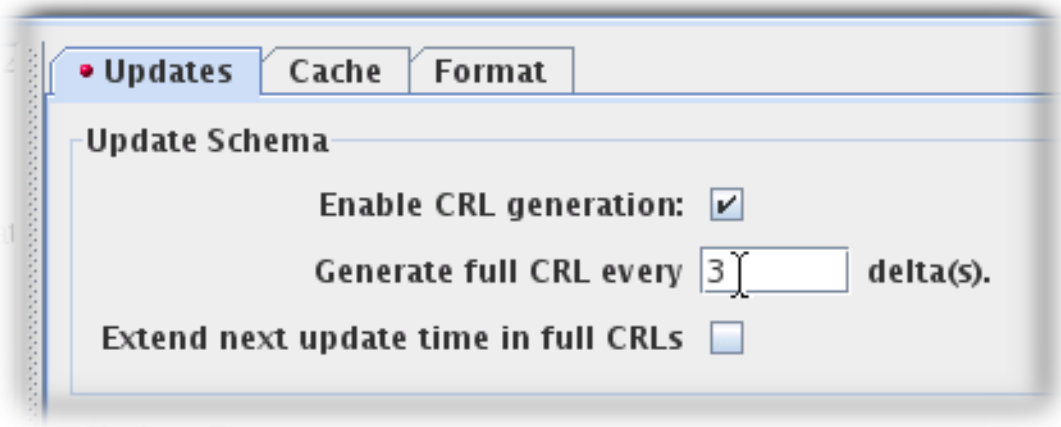
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、**Certificate Manager** フォルダーと **CRL Issuing Points** サブディレクトリーを展開します。

3. **MasterCRL** ノードを選択します。



4. **Generate full CRL every # delta(s)** フィールドに、必要な間隔を入力します。

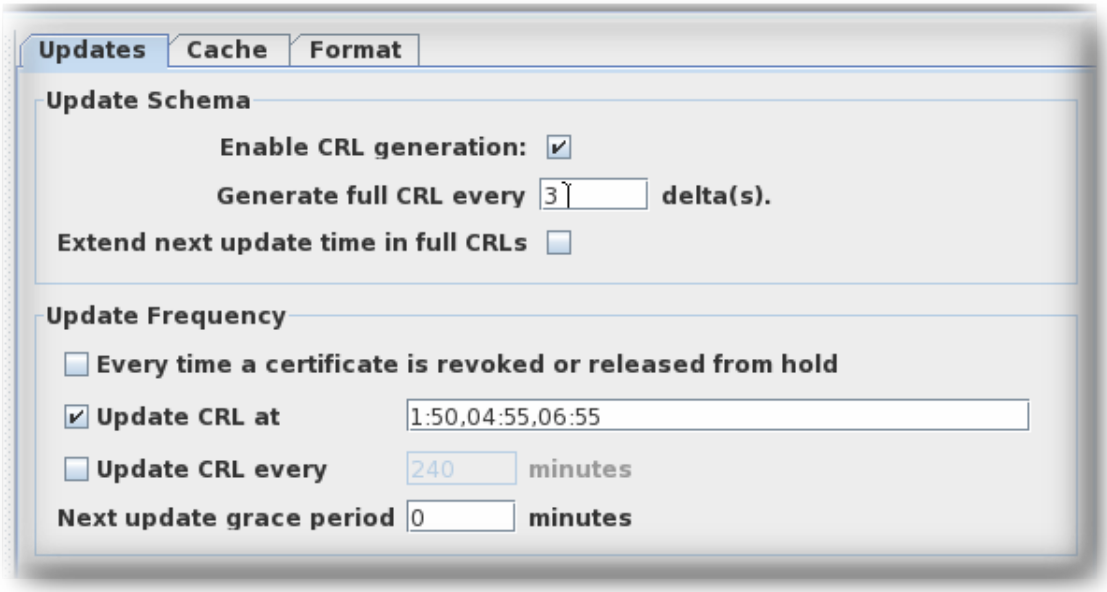


5. 証明書失効の機会、周期的な間隔、または更新が発生する時間を設定することにより、更新頻度を設定します。

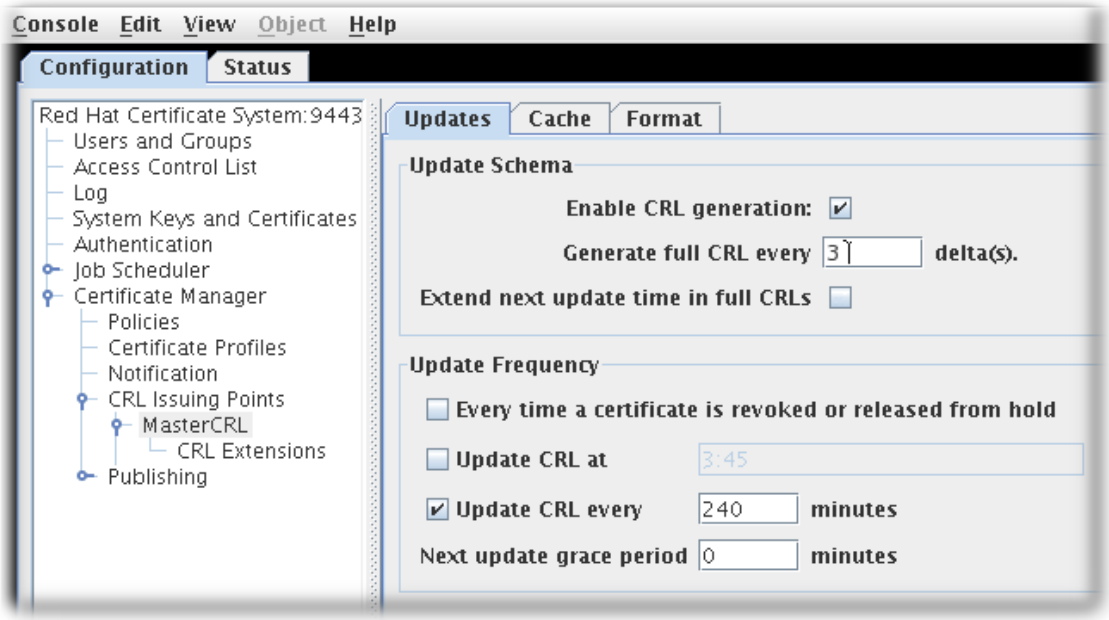
   - **Update CRL every time a certificate is revoked or released from hold** チェックボックスを選択します。**Update CRL every time a certificate is revoked or released from hold** オプションでも、2 つの **Grace period** 設定を入力する必要があります。これは既知の問題で、バグは Red Hat Bugzilla で追跡されています。

   - **Update CRL every time a certificate is revoked or released from hold** チェックボックスを選択します。

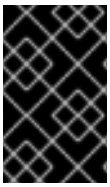
   - **Update CRL at** チェックボックスを選択し、**01:50,04:55,06:55** などの特定の時刻をコンマで区切って入力します。



- **Update CRL every** チェックボックスを選択し、**240** などの必要な間隔を入力します。



6. 変更を保存します。



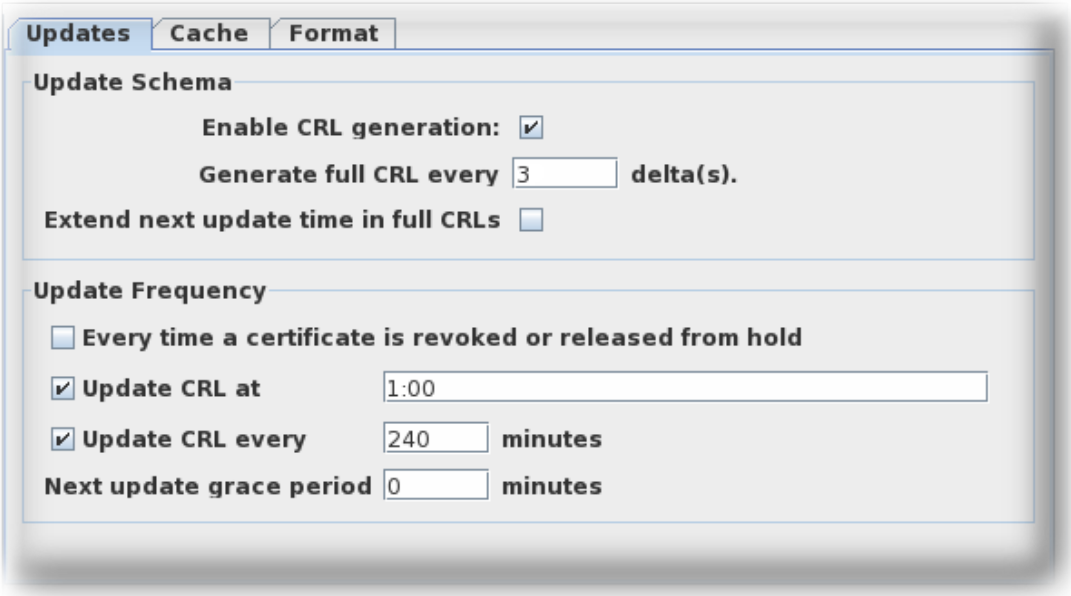
重要

Update CRL every time a certificate is revoked or released from hold オプションでも、2 つの **grace period** 設定を入力する必要があります。これは既知の問題で、バグは Red Hat Bugzilla で追跡されています。

注記

間隔ごとに CRL を更新するとドリフトが発生する場合があります。通常、ドリフトは手動更新と CA の再起動時に実行されます。

スケジュールのずれを防ぐには、**Update CRL at** チェックボックスを選択して値を入力します。間隔の更新は、24 時間ごとに **Update CRL at** 値と再同期します。



間隔で CRL を更新する場合は、**Update CRL at** 値は 1 つだけ受け入れられます。

6.4.2. 複数の日における CRL 生成スケジュールの設定

デフォルトで、CRL 生成のスケジュールは 24 時間に対応しています。また、デフォルトでは、フル CRL とデルタ CRL が有効になっている場合、1 つまたはすべてのデルタ CRL の代わりに、特定の間隔、つまり 3 回の更新ごとにフル CRL が発生します。

複数日にわたる CRL 生成スケジュールを設定するには、時間のリストでコンマを使用して同じ日の時間を区切り、セミコロンを使用して日を区切ります。

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

この例では、スケジュールの 1 日目の 01:00、03:00、および 18:00 と、スケジュールの 2 日目の 02:00、05:00、および 17:00 に CRL を更新します。3 日目にサイクルが再開します。

注記

セミコロンは新規日を示します。セミコロンで一覧を開始すると、CRL が生成されない最初の日になります。同様に、リストをセミコロンで終了すると、CRL が生成されないスケジュールに最終日が追加されます。2 つのセミコロンを合わせると、CRL が生成されない日になります。

デルタ更新とは独立してフル CRL 更新を設定するために、時間のリストは、完全な CRL 更新がいつ発生するかを示すアスタリスクが前に付いた時間値を受け入れます。

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

この例では、1 日目の 01:00、03:00、および 18:00 にデルタ CRL 更新を生成し、23:00 にフル CRL およびデルタ CRL の更新を生成します。2 日目では、デルタ CRL は 02:00、05:00、および 21:00 で更新されます。これは、フル CRL およびデルタ CRL の更新が 23:30 で行われます。3 日目にサイクルが再開します。

注記

コンソールでは、セミコロンとアスタリスクの両方の構文が機能します。

6.5. OCSP (ONLINE CERTIFICATE STATUS PROTOCOL) レスポンダーの使用

6.5.1. OCSP レスポンダーの設定

Online Certificate Status Manager の設定時にセキュリティードメイン内の CA が選択される場合は、OCSP サービスを設定する追加の手順は必要ありません。CA の CRL 公開は自動的に設定され、その署名証明書は Online Certificate Status Manager の証明書データベースで自動的に追加および信頼されます。ただし、セキュリティーのないドメイン CA を選択した場合は、Online Certificate Status Manager の設定後に OCSP サービスを手動で設定する必要があります。

注記

OCSP Manager が属するセキュリティードメイン内のすべての CA は、設定時に OCSP Manager によって自動的に信頼されるわけではありません。CA パネルで設定された CA の証明書チェーン内のすべての CA は、OCSP マネージャーによって自動的に信頼されます。セキュリティードメイン内にあるが証明書チェーンにはない他の CA は、手動で信頼させる必要があります。

セキュリティードメイン外の Certificate Manager に Online Certificate Status Manager を設定するには、次を行います。

1. OCSP レスポンダーに公開されるすべての CA に CRL を設定します。
2. OCSP サービスが処理するすべての CA で、公開を有効にし、パブリッシャーを設定し、公開ルールを設定します (*7章証明書および CRL の公開*)。Certificate Manager が LDAP ディレクトリーに公開され、Online Certificated Status Manager がそのディレクトリーから読み込むように設定している場合は、これは必要ありません。
3. 証明書プロファイルは、Certificate Manager が OCSP サービス要求をリッスンする場所を指す Authority Information Access 拡張機能を含むように設定する必要があります (「証明書マネージャーの内部 OCSP サービスの有効化」)。
4. OCSP Responder を設定します。
    - 失効情報ストア (「失効情報ストアの設定: 内部データベース」 および 「失効情報ストアの設定: LDAP ディレクトリー」) を設定します。
    - OCSP レスポンダー (「OCSP レスポンダーへの CA の特定」) へのすべての公開証明書マネージャーを特定します。
    - 必要に応じて、OCSP 署名証明書 (「CA 証明書の信頼設定の変更」) に署名した CA に信頼を設定します。
5. 設定後に両方のサブシステムを再起動します。

6. CA が OCSP レスポンダーに適切に接続されていることを確認します (「証明書マネージャーおよびオンライン証明書ステータスマネージャーの接続の確認」)。

6.5.2. OCSP レスポンダーへの CA の特定

CRL を Online Certificate Status Manager に公開するように CA を設定する前に、Online Certificate Status Manager の内部データベースに CA 署名証明書を保存することにより、CA を Online Certificate Status Manager に識別する必要があります。Certificate Manager は、この証明書に関連するキーペアの CRL を署名します。Online Certificate Status Manager は、保存した証明書に対して署名を検証します。

注記

Online Certificate Status Manager の設定時にセキュリティードメイン内の CA が選択されている場合は、CA を認識するように Online Certificate Status Manager を設定する手順が追加する必要はありません。CA 署名の証明書は自動的に追加され、Online Certificate Status Manager の証明書データベースで信頼されます。ただし、非セキュリティードメイン CA が選択されている場合は、Online Certificate Status Manager を設定した後、CA 署名証明書を証明書データベースに手動で追加する必要があります。

CRL を Online Certificate Status Manager に公開する CA の証明書チェーンをインポートする必要はありません。OCSP サービスに証明書チェーンが必要なのは、CA が CRL を公開するときに TLS 認証を介して Online Certificate Status Manager に接続する場合のみです。それ以外の場合は、Online Certificate Status Manager に完全な証明書チェーンは必要ありません。

ただし、Online Certificate Status Manager の証明書データベースには、CRL に署名した証明書 (CA 署名証明書または個別の CRL 署名証明書) が必要です。OCSP サービスは、CRL に署名した証明書を、証明書チェーンではなく、データベース内の証明書と比較することにより、CRL を検証します。ルート CA とその下位 CA の 1 つが CRL を Online Certificate Status Manager に公開する場合、Online Certificate Status Manager には両方の CA の CA 署名証明書が必要です。

CA が Online Certificate Status Manager に公開している証明書の署名に使用される CA または CRL 署名証明書をインポートするには、次の手順を実行します。

1. Certificate Manager の base-64 CA 署名証明書は、CA のエンドエンティティーページから取得します。

2. オンライン証明書ステータスマネージャーエージェントページを開きます。URL の形式は **https://hostname:SSLport/ocsp/agent/ocsp** です。

3. 左側のフレームで、**Add Certificate Authority** をクリックします。

4. フォームで、エンコードされた CA 署名証明書を **Base 64 encoded certificate (including the header and footer)** というラベルの付いたテキスト領域内に貼り付けます。

5. 証明書が正常に追加されたことを確認するには、左側のフレームで **List Certificate Authorities** をクリックします。

その結果、新しい CA に関する情報が表示されます。**This Update** フィールド、**Next Update**、および **Requests Served Since Startup** フィールドには、ゼロ (0) の値が表示されます。

6.5.2.1. 証明書マネージャーおよびオンライン証明書ステータスマネージャーの接続の確認

Certificate Manager を再起動すると、Online Certificate Status Manager の TLS ポートに接続しようとします。Certificate Manager が実際に Online Certificate Status Manager と通信したことを確認するに

は、**This Update** フィールドおよび **Next Update** フィールドを確認します。これらのフィールドは、CA が Online Certificate Status Manager と最後に通信したときの適切なタイムスタンプで更新する必要があります。クライアントが証明書失効リストのステータスに対して OCSP サービスにクエリーを試行していないため、**Requests Served Since Startup** フィールドにはゼロ (0) の値が表示されるはずです。

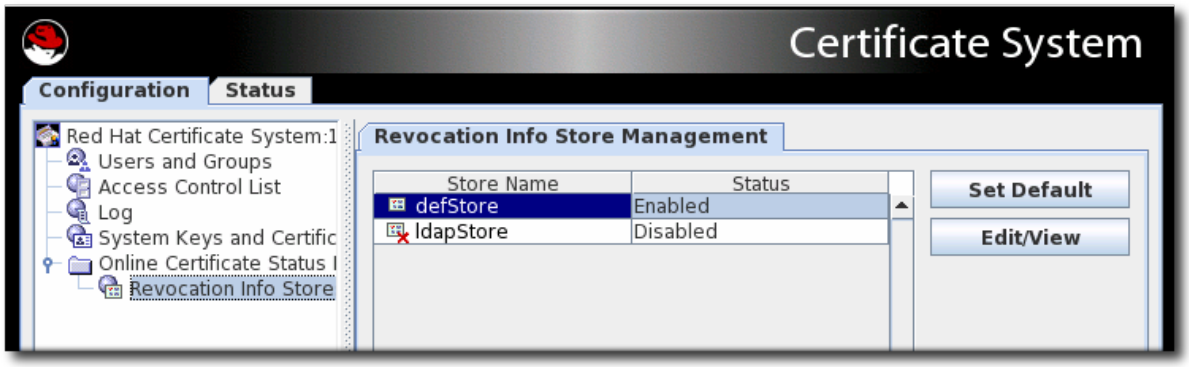
6.5.2.2. 失効情報ストアの設定: 内部データベース

Online Certificate Status Manager は各 Certificate Manager の CRL を内部データベースに保存し、これを CRL ストアとして使用し、証明書の失効ステータスを確認します。Online Certificate Status Manager が CRL を内部データベースに格納するために使用する設定を変更するには、以下を実行します。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

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

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。



右側のペインには、Online Certificate Status Manager が使用できる 2 つのリポジトリーが表示されます。デフォルトでは、内部データベースで CRL を使用します。

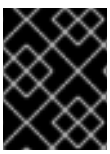
3. **defStore** を選択して **Edit/View** をクリックします。

4. **defStore** 値を編集します。

- **notFoundAsGood.**問題の証明書が CRL のいずれかに見つからない場合は、GOOD の OCSP 応答を返すように OCSP サービスを設定します。これを選択しないと、応答は UNKNOWN になり、クライアントが発生した場合にはエラーメッセージが表示されます。
- **byName.**OCSP レスポンダーは、応答を行う OCSP レスポンダーの ID を含む基本的な応答タイプのみをサポートします。基本応答タイプの ResponderID フィールドは、***ocsp.store.defStore.byName*** パラメーターの値により決定されます。***byName*** パラメーターが true である、または存在しない場合、OCSP 認証局署名証明書サブジェクト名は OCSP 応答の ResponderID フィールドとして使用されます。***byName*** パラメーターが false の場合、OCSP 認証局署名証明書キーハッシュは OCSP 応答の ResponderID フィールドになります。
- **includeNextUpdate.**次の CRL 更新時間のタイムスタンプが含まれます。

6.5.2.3. 失効情報ストアの設定: LDAP ディレクトリー

OCSP Manager はデフォルトでは CA CRL を内部データベースに保存しますが、代わりに LDAP ディレクトリーに公開された CRL を使用するよう設定することができます。



重要

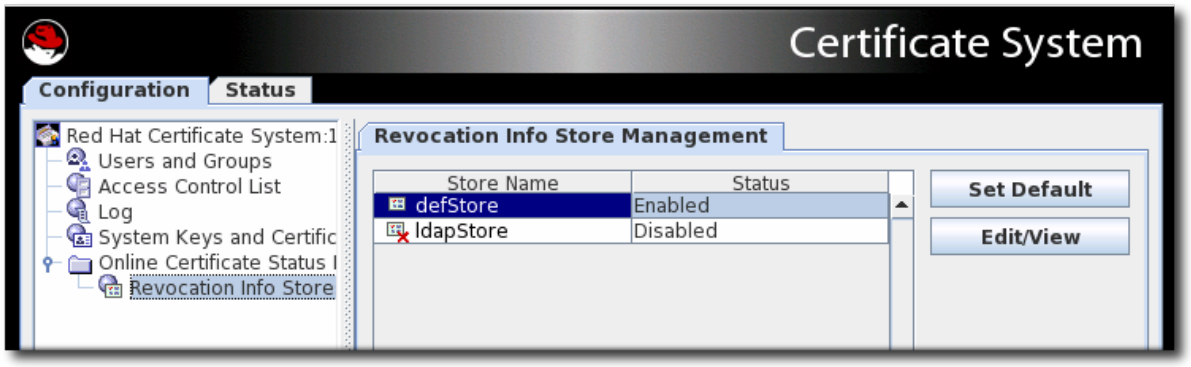
ldapStore メソッドが有効になっていると、OCSP ユーザーインターフェイスは証明書のステータスを確認しません。

LDAP ディレクトリーを使用するように Online Certificate Status Manager を設定するには、以下を実行します。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

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

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。

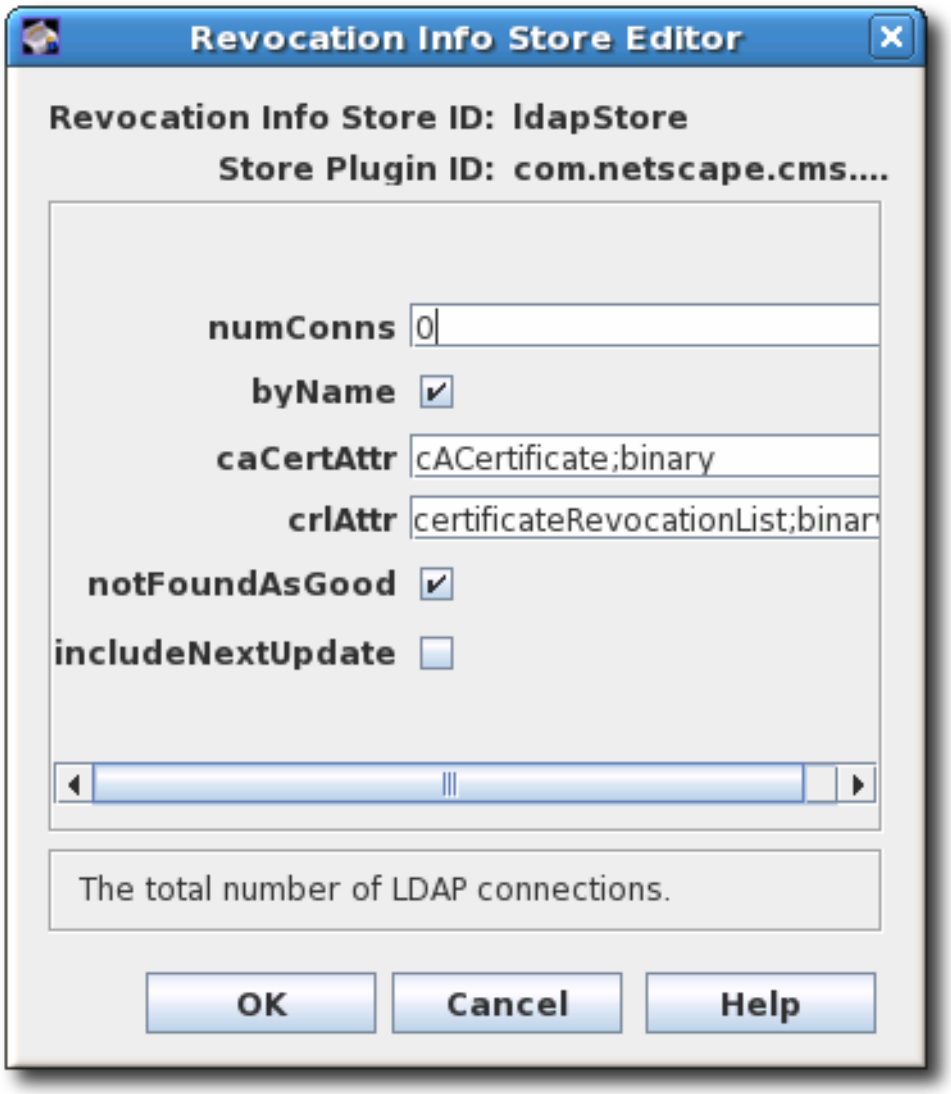


右側のペインには、Online Certificate Status Manager が使用できる 2 つのリポジトリーが表示されます。デフォルトでは、内部データベースで CRL を使用します。

3. LDAP ディレクトリーで CRL を使用するには、**Set Default** をクリックして **ldapStore** オプションを有効にします。

4. **ldapStore** を選択して **Edit/View** をクリックします。

5. **ldapStore** パラメーターを設定します。



- **numConns.**OCSP サービスがチェックする必要のある LDAP ディレクトリーの合計数。デフォルトでは、これは 0 に設定されます。この値を設定すると、対応する**host** フィールド、**port** フィールド、**baseDN** フィールド、および **refreshInSec** フィールドの数が表示されます。

- **host.**LDAP ディレクトリーの完全修飾 DNS ホスト名。

- **port。**LDAP ディレクトリーの TLS ポート以外のポート。

- **baseDN.**CRL の検索を開始する DN。たとえば、**O=example.com** です。

- **refreshInSec.**接続が更新される頻度。デフォルトは 86400 秒 (毎日) です。

- **caCertAttr.**デフォルト値である **cACertificate;binary** はそのままにしておきます。これは、Certificate Manager がその CA 署名証明書を公開する属性です。

- **crlAttr.**デフォルト値 **certificateRevocationList;binary** はそのままにしておきます。これは、Certificate Manager が CRL を公開する属性です。

- **notFoundAsGood.**問題の証明書が CRL のいずれかに見つからない場合は、GOOD の OCSP 応答を返すように OCSP サービスを設定します。これを選択しないと、応答は UNKNOWN になり、クライアントが発生した場合にはエラーメッセージが表示されます。

- **byName.**OCSP レスポンダーは、応答を行う OCSP レスポンダーの ID を含む基本的な応答タイプのみをサポートします。基本応答タイプの ResponderID フィールド

は、***ocsp.store.defStore.byName*** パラメーターの値により決定されます。***byName*** パラメーターが true である、または存在しない場合、OCSP 認証局署名証明書サブジェクト名は OCSP 応答の ResponderID フィールドとして使用されます。***byName*** パラメーターが false の場合、OCSP 認証局署名証明書キーハッシュは OCSP 応答の ResponderID フィールドになります。

- **includeNextUpdate.**Online Certificate Status Manager には、次の CRL 更新時間のタイムスタンプを含めることができます。

6.5.2.4. OCSP サービス設定のテスト

以下を実行して、Certificate Manager が OCSP 要求を適切に処理できるかどうかをテストします。

1. ブラウザーまたはクライアントで失効チェックをオンにします。
2. OCSP サービス用に有効になっている CA から証明書を要求します。
3. 要求を承認します。
4. ブラウザーまたはクライアントに証明書をダウンロードします。
5. CA がブラウザーまたはクライアントで信頼されていることを確認します。
6. Certificate Manager の内部 OCSP サービスのステータスを確認します。

   CA エージェントサービスページを開き、**OCSP サービス** のリンクを選択します。
7. 独立した Online Certificate Status Manager サブシステムをテストします。

   Online Certificate Status Manager エージェントサービスページを開き、**List Certificate Authorities** リンクをクリックします。

   このページには、CRL を Online Certificate Status Manager に公開するための設定された Certificate Manager に関する情報が表示されます。このページには、最後に起動した時点の Online Certificate Status Manager のアクティビティーも要約されています。
8. 証明書を取り消します。
9. ブラウザーまたはクライアントで証明書を確認します。サーバーは、証明書が取り消されたことを返す必要があります。
10. Certificate Manager の OCSP サービスステータスを再度チェックして、次のことが発生したことを確認します。

    - ブラウザーは OCSP クエリーを Certificate Manager に送信します。
    - Certificate Manager は OCSP の応答をブラウザーに送信します。
    - ブラウザーはその応答を使用して証明書を検証し、証明書を検証できなかったというステータスを返しました。
11. 独立した OCSP サービスサブシステムを再度チェックし、これらの問題が発生することを確認します。

    - 証明書マネージャーは、CRL を Online Certificate Status Manager に公開します。
    - ブラウザーは OCSP 応答を Online Certificate Status Manager に送信します。

- Online Certificate Status Manager は OCSP の応答をブラウザーに送ります。
- ブラウザーはその応答を使用して証明書を検証し、証明書を検証できなかったというステータスを返しました。

6.5.3. 問題のあるシリアル番号のレスポンスの設定

OCSP レスポンダーは、証明書が有効かどうかを判断する前に、証明書の失効ステータスと有効期限を確認します。デフォルトでは、OCSP は証明書の他の情報を検証しません。

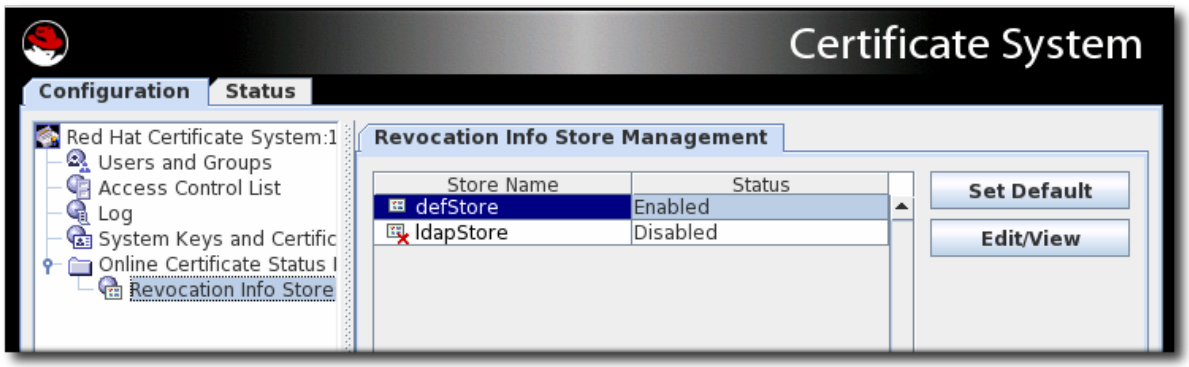
notFoundAsGood パラメーターは、OCSP が無効なシリアル番号で証明書を処理する方法を設定します。このパラメーターはデフォルトで有効になっています。つまり、証明書が不正なシリアル番号で存在する場合は、証明書が有効であれば、OCSP が証明書の **GOOD** のステータスを返します。

OCSP に、不正なシリアル番号と失効ステータスに基づいて証明書をチェックおよび拒否させるには、***notFoundAsGood*** 設定を変更します。この場合、OCSP は、間違ったシリアル番号を持つ証明書とともに **UNKNOWN** ステータスを返します。クライアントはエラーとして解釈し、それに応じて応答できます。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

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

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。



3. **defStore** を選択して **Edit/View** をクリックします。
4. ***notFoundAsGood*** 値を編集します。このチェックボックスを選択すると、証明書のシリアル番号が不正な場合でも OCSP が **GOOD** の値を返します。チェックボックスの選択を解除すると、OCSP は、**UNKNOWN** の値を送信します。クライアントはこれをエラーとして解釈できます。

5. OCSP Manager を再起動します。

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

6.5.4. 証明書マネージャーの内部 OCSP サービスの有効化

Certificate Manager には、OCSP 準拠のクライアントでビルトインの OCSP サービスがあり、Certificate Manager に、証明書の失効ステータスを直接問い合わせることができます。Certificate Manager がインストールされると、OCSP 署名証明書が発行され、OCSP サービスがデフォルトで有効になります。この OCSP 署名証明書は、OCSP サービスリクエストへのすべての応答に署名するために使用されます。内部 OCSP サービスは、Certificate Manager の内部データベースに格納されている証明書のステータスをチェックするため、このサービスを使用するように公開を設定する必要はありません。

クライアントは、Certificate Manager の TLS エンドエンティティーポートを介して OCSP サービスをクエリーできます。証明書失効ステータスをクエリーすると、Certificate Manager は証明書の内部データベースを検索し、そのステータスを確認してクライアントに応答します。Certificate Manager は発行されたすべての証明書のリアルタイムステータスであるため、失効確認の方法は最も正確です。

インストール時に、すべての CA ビルトイン OCSP サービスが有効になっている。ただし、このサービスを使用するには、CA が Authority Information Access 拡張で証明書を発行する必要があります。

1. CA のエンドエンティティーに移動します。以下に例を示します。

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

2. CA 署名証明書を探します。
3. 証明書で Authority Info Access 拡張を探し、**https://server.example.com:8443/ca/ocsp** などの **Location URIName** 値をメモします。
4. 登録プロファイルを更新して、Authority Information Access 拡張を有効にし、**Location** パラメーターを Certificate Manager の URI に設定します。証明書プロファイルの編集に関する詳細は、「証明書プロファイルの設定」 を参照してください。
5. CA インスタンスを再起動します。

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

6.5.5. OCSPClient プログラムを使用した OCSP リクエストの送信

OCSPClient プログラムは、OCSP リクエストの実行に使用できます。以下に例を示します。

```
]# 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 コマンドは、以下のコマンドラインオプションと共に使用できます。

表6.1 利用可能な OCSPClient オプション

オプション	説明
-d *database*	セキュリティーデータベースの場所 (デフォルト: 現行ディレクトリー)
-h *hostname*	OCSP サーバーのホスト名 (デフォルト: example.com)
-p *port*	OCSP サーバーのポート番号 (デフォルト: 8080)
-t *path*	OCSP サービスパス (デフォルト: /ocsp/ee/ocsp)
-c *nickname*	CA 証明書のニックネーム (デフォルト: CA 署名証明書)
-n *times*	送信番号 (デフォルトは 1)
--serial *serial_number*	チェックする証明書のシリアル番号
--input *input_file*	DER でエンコードされた OCSP 要求が含まれる入力ファイル
--output *output_file*	DER でエンコードされた OCSP 応答を保存する出力ファイル
-v, --verbose	詳細モードで実行

オプション	説明
--help	ヘルプメッセージを表示

6.5.6. GET メソッドを使用した OCSP リクエストの送信

RFC 6960 で説明されているように、255 バイト未満の OCSP 要求は、GET メソッドを使用して Online Certificate Status Manager に送信できます。GET 経由で OCSP 要求を送信するには、以下のコマンドを実行します。

1. クエリーされるステータスで、証明書の OCSP 要求を生成します。以下に例を示します。

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

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. Web ブラウザーのアドレスバーに URL を貼り付けて、ステータス情報を返します。ブラウザーが OCSP 要求を処理できるようにする必要があります。

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

3. OCSP Manager は、ブラウザーが解釈できる証明書のステータスを返します。設定可能なステータスは GOOD、REVOKED、および UNKNOWN です。

あるいは、**curl** などのツールを使用して、要求と **openssl** を使用して応答を解析して、コマンドラインから OCSP を実行します。以下に例を示します。

1. クエリーされるステータスで、証明書の OCSP 要求を生成します。以下に例を示します。

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

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. **curl** を使用して、OCSP 要求を送信するために OCSP Manager に接続します。

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

3. **openssl** を使用して応答を解析します。

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

パート III. CA サービスを管理するための追加設定

第7章 証明書および CRL の公開

注記

TMS 上の本セクションにある機能は、評価でテストされていません。本セクションは参照用途としてのみ提供されています。

Red Hat Certificate System には、Certificate Manager 用のカスタマイズ可能な公開フレームワークが含まれており、証明書機関は、証明書、証明書失効リスト (CRL)、およびその他の証明書関連オブジェクトを、サポートされているリポジトリー (LDAP 準拠のディレクトリー、フラットファイル、およびオンライン検証機関) に有効にします。本章では、証明書および CRL をファイル、ディレクトリー、および Online Certificate Status Manager に公開するように Certificate Manager を設定する方法を説明します。

パブリッシュを設定する一般的なプロセスは次のとおりです。

1. ファイル、LDAP ディレクトリー、または OCSP レスポンダーへの公開を設定します。

   使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

2. ルールを設定して、どの証明書がその場所に公開されるかを決定します。証明書または CRL が一致するすべてのルールがアクティブ化されるため、ファイルベースのルールとディレクトリーベースのルールを一致させることにより、同じ証明書をファイルと LDAP ディレクトリーに公開できます。

   ルールは、各オブジェクトタイプ (CA 証明書、CRL、ユーザー証明書、およびクロスペアの証明書) に設定できます。使用されていないルールをすべて無効にします。

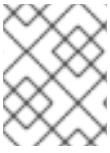
3. CRL を設定します。CRL は公開前に設定する必要があります。*6章 証明書の取り消しおよび CRL 発行* を参照してください。

4. パブリッシャー、マッパー、およびルールの設定後に公開を有効にします。公開が有効になると、サーバーはすぐに公開を開始します。パブリッシャー、マッパー、およびルールが完全に設定されていない場合は、パブリッシュが正しく機能しない可能性があります。

7.1. 公開の概要

証明書システムは、ファイルまたは LDAP ディレクトリーに証明書を公開したり、CRL をファイル、LDAP ディレクトリー、OCSP レスポンダーに公開したりできます。

柔軟性を高めるために、特定のタイプの証明書または CRL を単一の形式または 3 つすべての形式で公開できます。たとえば、CA 証明書はディレクトリーにのみ公開され、ファイルには公開されず、ユーザー証明書はファイルとディレクトリーの両方に公開できます。



注記

OCSP レスポンダーは CRL に関する情報のみを提供します。証明書は OCSP レスポンダーに公開されません。

証明書ファイルと CRL ファイルに異なる公開場所を設定でき、さまざまな種類の証明書ファイルや異なるタイプの CRL ファイルとの間で異なる公開場所を設定することができます。

同様に、異なるタイプの証明書や異なるタイプの CRL をディレクトリー内の異なる場所に公開できます。たとえば、所属企業の West Coast 部門からの証明書は、ディレクトリーの 1 つのブランチで公開することができますが、East Coast 部門のユーザーの証明書をディレクトリー内の他のブランチに公開することができます。

公開が有効になっている場合、証明書または CRL が発行、更新、または取り消されるたびに、公開システムが呼び出されます。証明書または CRL はルールによって評価され、ルールのタイプおよび述語と一致するかどうかを確認します。タイプは、オブジェクトが CRL、CA 証明書、またはその他の証明書であるかどうかを指定します。述語は、評価されるオブジェクトのタイプに対してさらに基準を設定します。たとえば、ユーザー証明書を指定するか、West Coast ユーザー証明書を指定できます。述語を使用するには、公開ルールの述語フィールドに値を入力する必要があります。また、対応する値 (形式は多少異なります) を証明書または証明書要求に含める必要があります。証明書または証明書要求の値は、証明書のタイプなどの証明書の情報から取得することも、要求フォームに配置された非表示の値から取得することもできます。述語が設定されていない場合は、そのタイプのすべての証明書が一致することが考慮されます。たとえば、**CRL** がタイプとして設定されている場合、すべての CRL がルールに一致します。

マッチするすべてのルールは、そのルールで指定された方法および場所に従って証明書または CRL を公開します。指定された証明書または CRL は、ルール、複数のルール、またはすべてのルールに一致しません。公開システムは、発行されたすべての証明書と CRL をすべてのルールと照合しようとします。

ルールがマッチすると、そのルールに関連するパブリッシャーに指定されたメソッドおよび場所に従って、証明書または CRL が公開されます。たとえば、ルールがユーザーに発行されたすべての証明書と一致して、ルールにその場所 **/etc/CS/certificates** のファイルに公開する発行者がある場合、証明書はファイルとしてその場所に公開されます。別のルールが、ユーザーに発行されたすべての証明書に一致し、そのルールに LDAP 属性 **userCertificate;binary** 属性に公開するパブリッシャーがある場合、証明書は、ユーザーのエントリーのこの属性で LDAP 公開が有効になったときに指定されたディレクトリーに発行されます。

ファイルに公開するように指定するルールの場合、証明書または CRL が古くなったディレクトリーに新しいファイルが作成されます。

LDAP ディレクトリーに公開するように指定するルールの場合、指定された属性に、証明書または CRL がディレクトリーに指定されたエントリーに公開されます。CA は、公開された証明書または CRL 属性の値を後続の証明書または CRL で上書きします。簡単に言うと、すでに公開されている既存の証明書または CRL は、次の証明書または CRL に置き換えられます。

Online Certificate Status Manager への公開を指定するルールの場合、CRL はこのマネージャーに公開されます。証明書は Online Certificate Status Manager に公開されません。

LDAP 公開の場合は、ユーザーのエントリーの場所を決定する必要があります。マッパーは、公開するエントリーを決定するために使用されます。マッパーには、エントリーの正確な DN、証明書から取得した DN を作成するための情報を関連付けた変数、あるいはディレクトリーを検索してエントリー内の一意の属性や属性のセットを検索し、エントリーの正しい DN を確認するための十分な情報を含めることができます。

証明書が取り消されると、サーバーは公開ルールを使用して、LDAP ディレクトリーまたはファイルシステムから対応する証明書を見つけて削除します。

証明書の有効期限が切れると、サーバーは設定されたディレクトリーからその証明書を削除できます。サーバーはこれを自動的に実行しません。適切なジョブを実行するようにサーバーを設定する必要があります。

公開の設定には、パブリッシャー、マッパー、およびルールを設定する必要があります。

7.1.1. パブリッシャー

パブリッシャー は、証明書と CRL が公開される場所を指定します。ファイルに公開する場合、パブリッシャーはファイルシステムの公開ディレクトリーを指定します。LDAP ディレクトリーに公開する場合、発行者は証明書または CRL を格納するディレクトリーの属性を指定します。マッパーは、エントリーの DN を決定するために使用されます。DN ごとに、その DN を導出するための異なる式が設定されます。LDAP 公開が有効であるときに LDAP ディレクトリーの場所が指定されます。OCSP レスポンダーに CRL を公開する場合、パブリッシャーは Online Certificate Status Manager のホスト名と URI を指定します。

7.1.2. マッパー

マッパー は LDAP 公開でのみ使用されます。マッパーは、証明書または証明書要求からの情報に基づいて、エントリーの DN を構築します。サーバーには、証明書のサブジェクト名および証明書要求の情報があり、この情報を使用してそのエントリーの DN を作成する方法を把握する必要があります。マッパーは、利用可能な情報を DN、またはディレクトリー内で検索してエントリーの DN を取得できる一意の情報に変換するための式を提供します。

7.1.3. ルール

ファイル、LDAP、および OCSP 公開の **ルール** は、証明書または CRL を公開するかどうかとその方法をサーバーに指示します。ルールは、最初に、ルールのタイプと述語を設定することにより、公開されるもの、特定の特性に一致する証明書または CRL を定義します。次に、ルールは、パブリッシャーに関連付けられ、LDAP 公開の場合はマッパーに関連付けられることにより、公開方法と場所を指定します。

ルールは、PKI デプロイメントに必要なだけ単純または複雑にすることができ、さまざまなシナリオに対応するのに十分な柔軟性があります。

7.1.4. ファイルへの公開

サーバーは、証明書と CRL をフラットファイルに公開できます。フラットファイルは、リレーショナルデータベースなどの任意のリポジトリーにインポートできます。サーバーが証明書および CRL をファイルに公開するように設定されている場合、公開ファイルは DER でエンコードされたバイナリーブロブ、base-64 でエンコードされたテキストブロブ、またはその両方になります。

- サーバーの問題の各証明書について、DER でエンコードされた形式または base-64 でエンコードされた形式で、証明書が含まれるファイルを作成します。各ファイルには **cert-serial_number.der** または **cert-serial_number.b64** という名前が付けられます。**serial_number** は、ファイルに含まれる証明書のシリアル番号です。たとえば、シリアル番号が **1234** の DER でエンコードされた証明書のファイル名は、 **cert-1234.der** です。

- サーバーが CRL を生成するたびに、DER でエンコードされた形式または base-64 でエンコードされた形式で新しい CRL を含むファイルが作成されます。各ファイルの名前は、形式に応じて、**issuing_point_name-this_update.der** または **issuing_point_name-this_update.b64** のいずれかになります。**issuing_point_name** は CRL を公開する CRL 発行ポイントを識別します。**this_update** は、ファイルに含まれる CRL のタイム依存更新値から取得する値を指定します。たとえば、値が **This Update: Friday January 28 15:36:00 PST 2021** の DER でエンコードされた CRL のファイル名は **MasterCRL-20210128-153600.der** です。

7.1.5. OCSP 公開

Certificate System OCSP サービスには、Certificate Manager の内部サービスと Online Certificate Status Manager の 2 つの形式があります。内部サービスは、Certificate Manager の内部データベースを確認して、証明書のステータスを報告します。内部サービスは公開用に設定されていません。内部

データベースに格納されている証明書を使用して、証明書のステータスを判別します。Online Certificate Status Manager は、Certificate Manager によって送信される CRL を確認します。パブリッシャーは、CRL が送信される場所ごとに設定され、送信される各タイプの CRL に対して 1 つのルールが設定されます。

両方の OCSP サービスの詳細は、「OCSP (Online Certificate Status Protocol) レスポンダーの使用」を参照してください。

7.1.6. LDAP 公開

LDAP の公開 では、サーバーは LDAP または LDAPS を使用して証明書、CRL、およびその他の証明書関連のオブジェクトをディレクトリーに公開します。公開するディレクトリーのブランチは、*公開ディレクトリー* と呼ばれます。

- サーバーが発行する証明書ごとに、ユーザーのエントリーの指定された属性に DER エンコード形式の証明書を含むブロブが作成されます。証明書は DER でエンコードされたバイナリーブロブとして公開されます。

- サーバーが CRL を生成するたびに、CA のエントリーの指定された属性で、DER でエンコードされた形式で新しい CRL を含むブロブを作成します。

LDAP プロトコルまたは TLS (LDAPS) プロトコルを使用して、証明書および CRL を LDAP 準拠のディレクトリーに公開し、アプリケーションは HTTP 経由で証明書および CRL を取得できます。HTTP で証明書および CRL の取得をサポートすると、一部のブラウザーはサーバーから通常の更新を受け取るディレクトリーから最新の CRL を自動的にインポートできます。ブラウザーは CRL を使用してすべての証明書を自動的にチェックし、証明書が取り消されていないことを確認できます。

LDAP 公開が機能するには、ユーザーエントリーが LDAP ディレクトリーに存在する必要があります。

サーバーと公開ディレクトリーが何らかの理由で同期しなくなった場合は、特権ユーザー (管理者とエージェント) も手動で公開プロセスを開始できます。手順は、「ディレクトリーでの CRL の手動による更新」を参照してください。

7.2. ファイルへの公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

ファイルへの公開は、CRL または証明書を特定のホスト上のテキストファイルに公開するだけです。

パブリッシャーは、発行場所ごとに作成および設定する必要があります。パブリッシャーは、ファイルに公開するために自動的に作成されません。すべてのファイルを単一の場所に公開するには、パブリッシャーを 1 つ作成します。異なる場所に公開するには、各場所にパブリッシャーを作成します。場所には、ユーザー証明書などのオブジェクトタイプ、または West Coast ユーザー証明書などのオブジェクトタイプのサブセットを含めることができます。

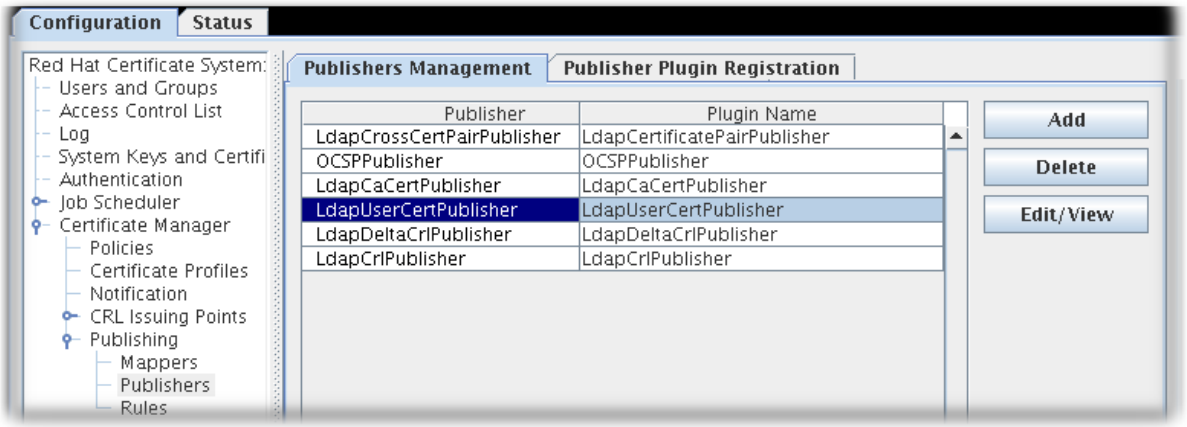
ファイルに公開するためのパブリッシャーを作成するには、以下の手順を実施します。

1. Certificate Manager コンソールにログインします。

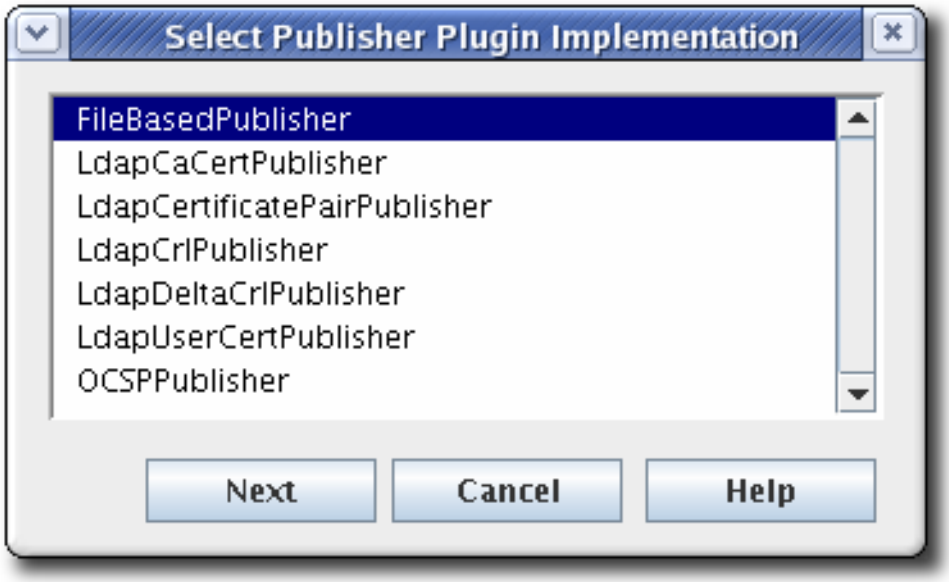
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Publishers** を選択します。

   設定されたパブリッシャーインスタンスを一覧表示する **Publishers Management** タブが右側で開きます。

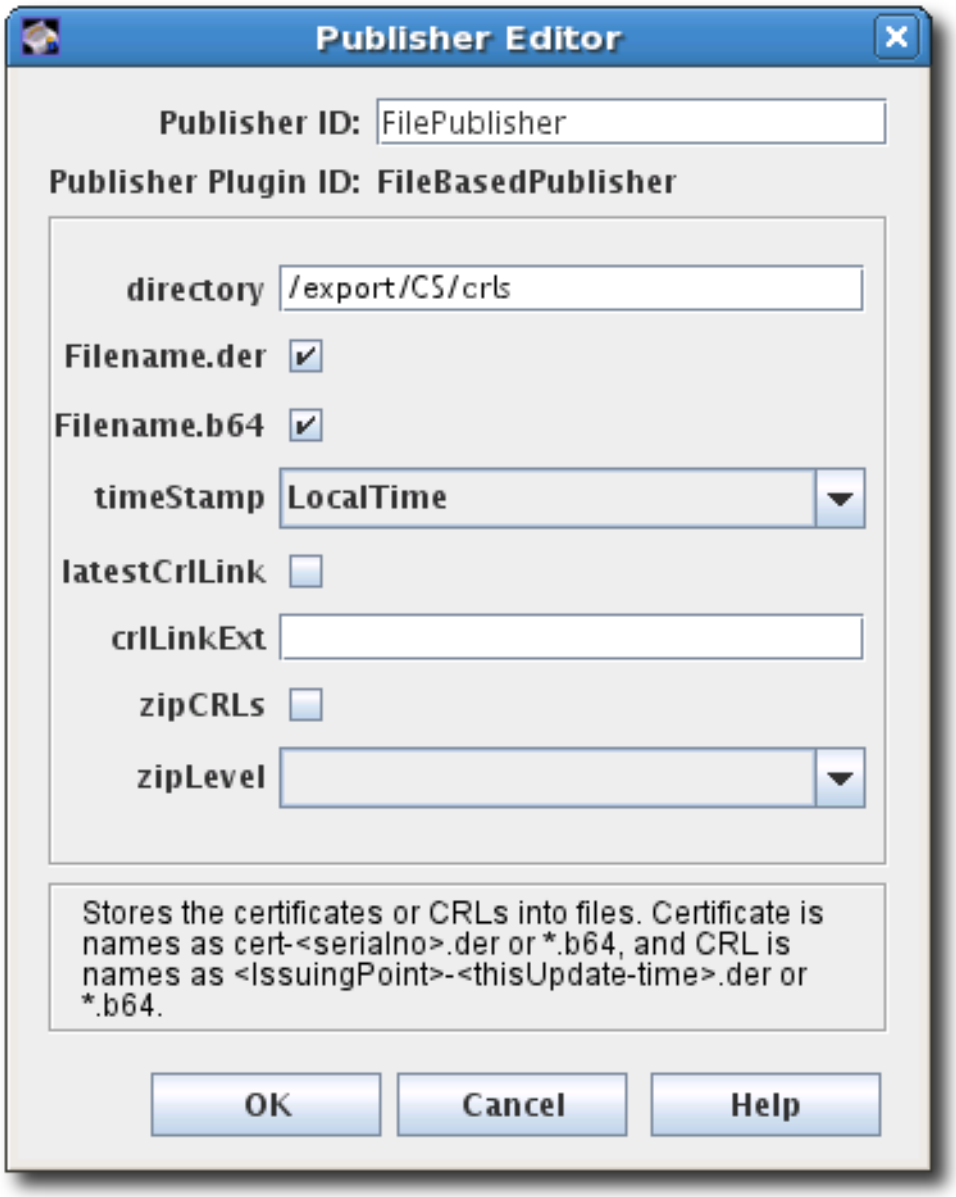


3. **Add** をクリックして、**Select Publisher Plug-in Implementation** ウィンドウを開きます。これには、登録済みのパブリッシャーモジュールを一覧表します。



4. **FileBasedPublisher** モジュールを選択して、エディターウィンドウを開きます。

   これは、Certificate Manager が証明書をファイルに公開し、CRL をファイルに公開できるようにするモジュールです。



5. 証明書を公開するための情報を設定します。

    - **PublishCertsToFile** などの空白のない英数字の発行側の ID
    - Certificate Manager がファイルを公開する必要のあるディレクトリーへのパス。パスは絶対パスを指定でき、Certificate System インスタンスのディレクトリーに相対することもできます。たとえば、**/export/CS/certificates** です。
    - DER でエンコードされたファイル、base-64 でエンコードされたファイル、またはその両方のチェックボックスを選択して公開するファイルタイプ。
    - CRL の場合は、タイムスタンプの形式です。公開される証明書にはファイル名にシリアル番号が含まれ、CRL はタイムスタンプを使用します。
    - CRL の場合、最新の CRL に移動するためにファイルにリンクを生成するかどうか。有効にすると、リンクは、拡張機能で使用する CRL 発行ポイントの名前が **crlLinkExt** フィールドに指定されると想定します。
    - CRL の場合、CRL を圧縮 (zip) するかどうか、および使用する圧縮レベル。

パブリッシャーを設定した後、「ルールの作成」 の説明に従って、発行された証明書と CR L のルールを設定します。

7.3. OCSP への公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

OCSP Manager への公開は、クライアント検証のために CRL を特定の場所に公開することです。

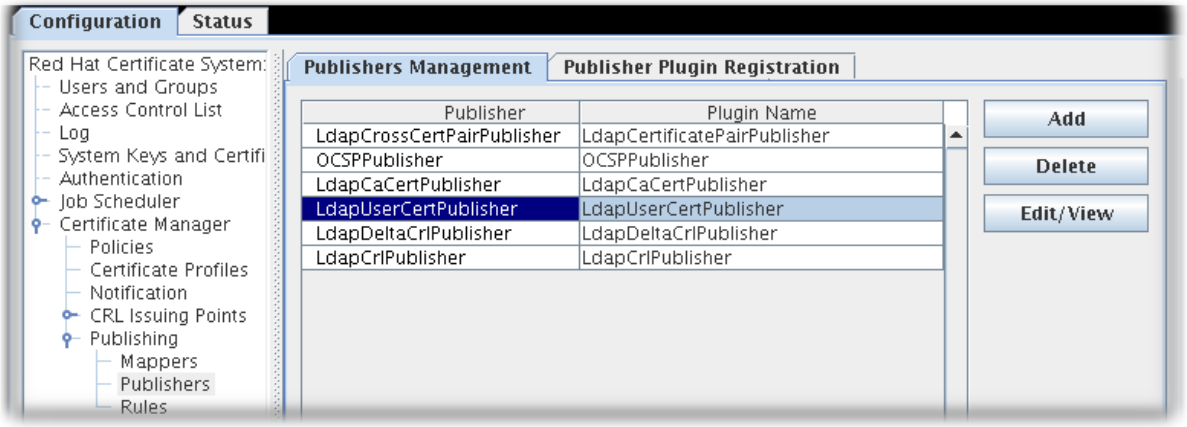
パブリッシャーは、公開場所ごとに作成および設定する必要があります。パブリッシャーは、OCSP レスポンダーに公開するために自動的に作成されません。単一のパブリッシャーを作成して、すべての場所を 1 つの場所に公開するか、CRL を公開するすべての場所のパブリッシャーを作成します。各場所には、さまざまな種類の CRL を含めることができます。

7.3.1. クライアント認証を使用した OCSP への公開の有効化

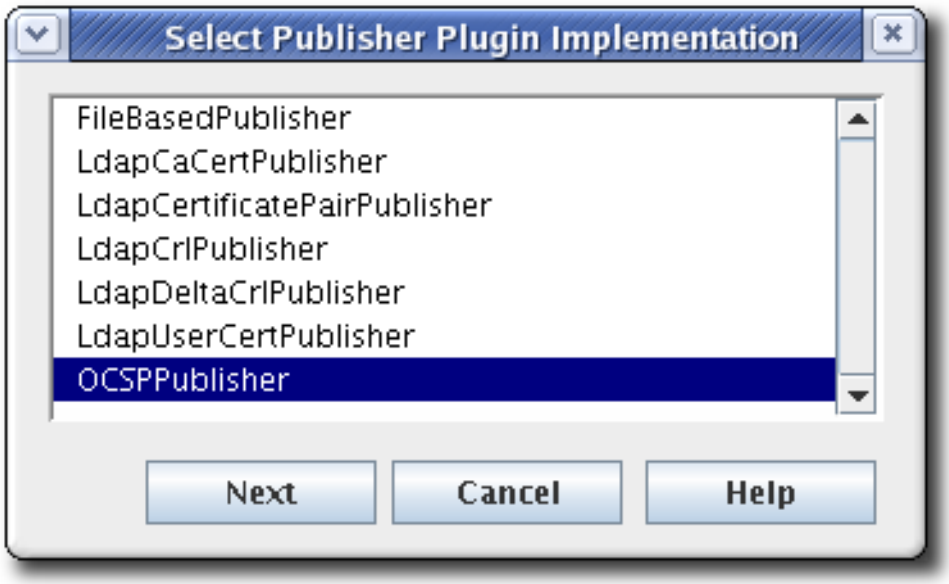
1. Certificate Manager コンソールにログインします。

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

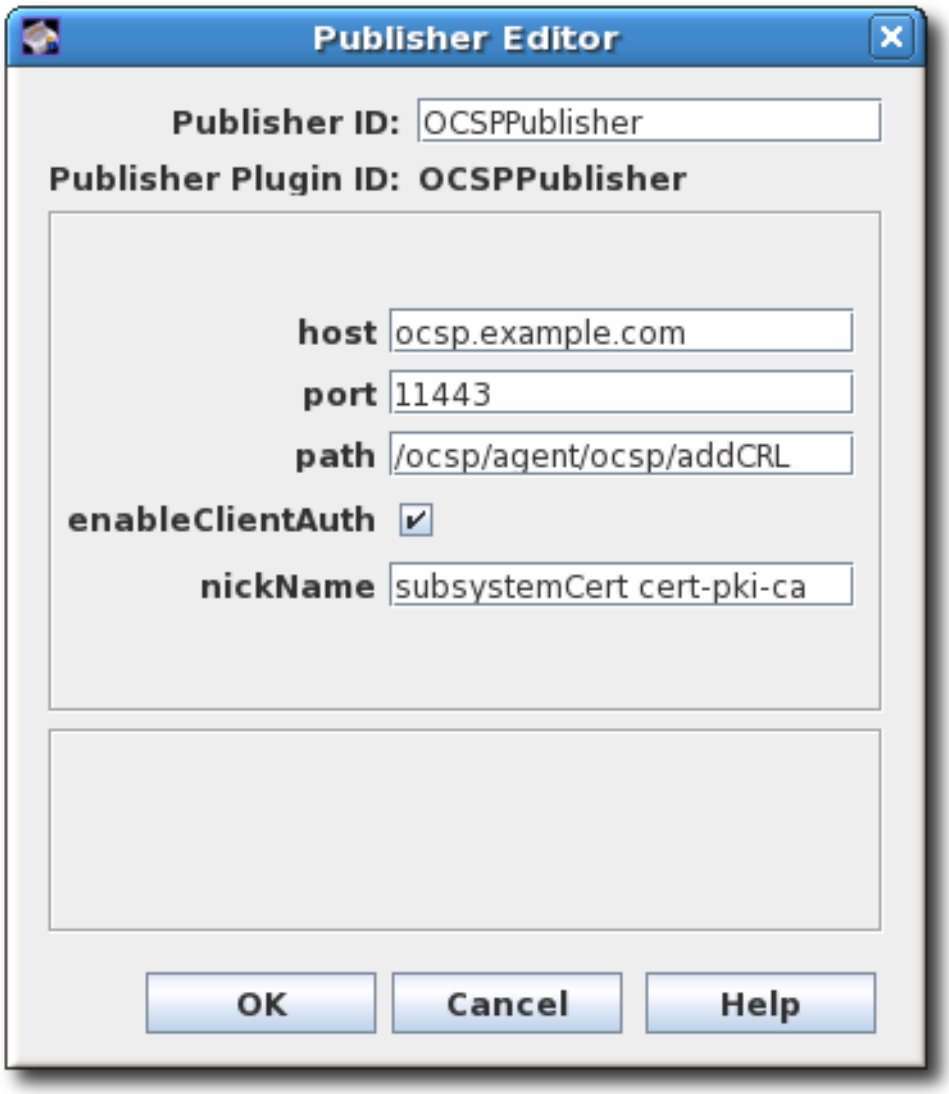
2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Publishers** を選択します。



3. **Add** をクリックして、**Select Publisher Plug-in Implementation** ウィンドウを開きます。これには、登録済みのパブリッシャーモジュールを一覧表します。



4. **OCSPPublisher** モジュールを選択し、エディターウィンドウを開きます。これは、Certificate Manager が CRL を Online Certificate Status Manager に公開できるようにするパブリッシャーモジュールです。



- パブリッシャー ID は、**PublishCertsToOCSP** のように、スペースのない英数字の文字列である必要があります。

- **host** は、**ocspResponder.example.com** または IPv4 または IPv6 アドレスなどの完全修飾ドメイン名を使用できます。
- デフォルトのパスは、**/ocsp/agent/ocsp/addCRL** のように CRL を送信するディレクトリーです。
- クライアント認証が使用されている (**enableClientAuth** が選択されている) 場合は、**nickname** フィールドに認証に使用する証明書のニックネームを指定します。この証明書は OCSP セキュリティーデータベースに存在している必要があります。これは通常 CA サブシステム証明書です。

5. OCSP Manager で CA のユーザーエントリーを作成します。ユーザーは、新しい CRL を送信するときに OCSP への認証に使用されます。必要なものは 2 つあります。
   - **CA-hostname-EEport** などの CA サーバーの後に OCSP ユーザーエントリーに名前を付けます。
   - パブリッシャー設定で指定された証明書を、OCSP ユーザーアカウントのユーザー証明書として使用します。通常、これは CA のサブシステム証明書です。

   サブシステムユーザーの設定については、「ユーザーの作成」 で説明されています。

パブリッシャーを設定した後、「ルールの作成」 の説明に従って、発行された証明書と CRL のルールを設定します。

7.4. LDAP ディレクトリーへの公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

LDAP 公開の設定は、ディレクトリーを設定するための追加のステップを除き、他の公開手順と似ています。

1. 公開される証明書の Directory Server を設定します。特定の属性をエントリーに追加し、ID をバインドし、認証方法を設定する必要があります。
2. 公開された各オブジェクトのパブリッシャーを設定します (CA 証明書、クロスペア証明書、CRL、およびユーザー証明書)。パブリッシャーは、オブジェクトを格納する属性を宣言します。デフォルトで設定される属性は、各オブジェクトタイプを保存する X.500 標準属性です。この属性はパブリッシャーで変更できますが、通常は LDAP パブリッシャーを変更する必要はありません。
3. エントリーの DN が証明書のサブジェクト名から派生できるようにマッパーを設定します。通常、CA 証明書、CRL、およびユーザー証明書にを設定する必要はありません。証明書タイプに複数のマッパーを設定できます。これは、たとえば、ディレクトリーツリーの異なる部分にある会社の異なる部門の 2 組のユーザーの証明書を公開する場合に役立ちます。グループごとにマッパーが作成され、ツリーの異なるブランチを指定します。

   マッパーの設定に関する詳細は、「マッパーの作成」 を参照してください。
4. 「ルールの作成」 で説明されているように、パブリッシャーをマッパーに接続するルールを作成します。
5. 「公開の有効化」 の説明に従って、パブリッシュを有効にします。

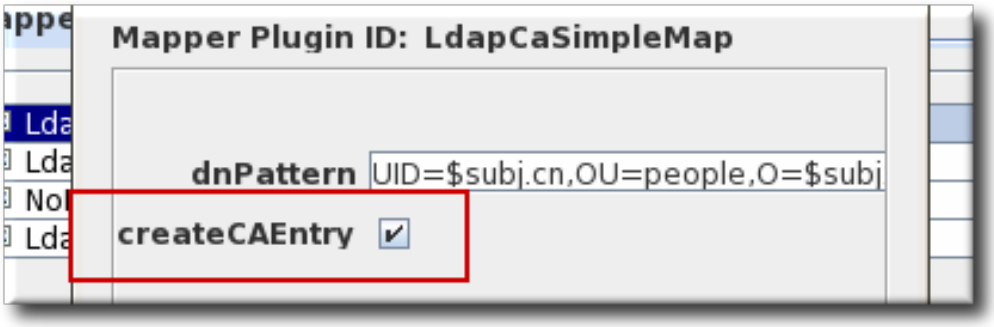
7.4.1. LDAP ディレクトリーの設定

証明書および CRL を公開する前に、Directory Server がパブリッシュシステムと連携するように設定する必要があります。つまり、ユーザーエントリーには証明書情報を受信できる属性が必要で、CRL を表すためにエントリーを作成する必要があります。

1. CA のエントリーを設定します。Certificate Manager が CA 証明書および CRL を公開するには、ディレクトリーには CA のエントリーが含まれている必要があります。

ヒント

LDAP 公開が設定されている場合、Certificate Manager はディレクトリー内の CA のエントリーを自動的に作成または変換します。このオプションは CA マッパーインスタンスおよび CRL マッパーインスタンスの両方で設定され、デフォルトで有効になっています。ディレクトリーが Certificate Manager がディレクトリーでエントリーを作成しないようにする場合は、これらのマッパーインスタンスでこのオプションを無効にし、CA を手動でディレクトリーに追加します。



CA のエントリーをディレクトリーに追加する場合は、CA の DN に基づいてエントリータイプを選択します。

- CA の DN が **cn** コンポーネントで開始する場合は、CA の新規 **person** エントリーを作成します。別のタイプのエントリーを選択すると、**cn** コンポーネントが指定されない場合があります。
- CA の DN が **ou** コンポーネントで開始する場合は、CA の新規 **organizationalunit** エントリーを作成します。

このエントリーは、**pkiCA** または **certificationAuthority** オブジェクトクラスにはありません。証明書マネージャーは、このエントリーを CA の署名証明書を公開して、**pkiCA** または **certificationAuthority** オブジェクトクラスに自動的に変換します。

注記

pkiCA オブジェクトクラスは RFC 4523 に定義されています が、**certificationAuthority** オブジェクトクラスは (obsolete) RFC 2256 に定義されています。Directory Server が使用するスキーマ定義に応じて、いずれかのオブジェクトクラスは受け入れ可能です。場合によっては、両方のオブジェクトクラスを同じ CA エントリーに使用できます。

ディレクトリーエントリーの作成に関する詳細は、Red Hat Directory Server のドキュメントを参照してください。

2. CA およびユーザーディレクトリーエントリーに正しいスキーマ要素を追加します。

Certificate Manager が証明書と CRL をディレクトリーに公開できるようにするには、特定の属性およびオブジェクトクラスで設定する必要があります。

オブジェクトタイプ	スキーマ	理由
エンドエンティティー証明書	userCertificate;binary (属性)	これは、証明書マネージャーが証明書をパブリッシュする属性です。 これは多値の属性で、各値は DER でエンコードされたバイナリー X.509 証明書です。**inetOrgPerson** という名前の LDAP オブジェクトクラスによりこの属性が許可されます。**strongAuthenticationUser** オブジェクトクラスはこの属性を許可し、他のオブジェクトクラスと組み合わせて、証明書を他のオブジェクトクラスのディレクトリーエントリーに公開できるようにすることができます。Certificate Manager は、このオブジェクトクラスを対応する Directory Server のスキーマテーブルに自動的に追加しません。 見つかったディレクトリーオブジェクトが **userCertificate;binary** 属性を許可しないと、証明書の追加または削除に失敗します。
CA 証明書	caCertificate;binary (属性)	これは、証明書マネージャーが証明書をパブリッシュする属性です。 Certificate Manager は、サーバーの起動時に独自の CA ディレクトリーエントリーに独自の CA 証明書を公開します。エントリーは、Certificate Manager の発行者名に対応します。 これは、**pkiCA** または **certificationAuthority** オブジェクトクラスの必須の属性です。Certificate Manager は、CA のディレクトリーエントリーを見つけると、このオブジェクトクラスを CA のディレクトリーエントリーに追加します。

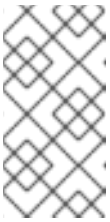
オブジェクトタイプ	スキーマ	理由
CRL	certificateRevocationList;binary (属性)	これは、Certificate Manager が CRL を公開する属性です。 Certificate Manager は、CRL を独自の LDAP ディレクトリーエントリーに公開します。エントリーは、Certificate Manager の発行者名に対応します。 これは、**pkiCA** または **certificationAuthority** オブジェクトクラスの属性です。属性の値は DER でエンコードされたバイナリー X.509 CRL です。CA のエントリーには、エントリーに CRL を公開するために、**pkiCA** または **certificationAuthority** オブジェクトクラスがすでに含まれている必要があります。
デルタ CRL	deltaRevocationList;binary (属性)	これは、Certificate Manager が証明書を公開する属性です。Certificate Manager は、フル CRL とは別に、デルタ CRL を独自の LDAP ディレクトリーエントリーに公開します。デルタ CRL エントリーは、Certificate Manager の発行者名に対応します。 この属性は、**deltaCRL** または **certificationAuthority-V2** オブジェクトクラスに属します。属性の値は DER でエンコードされたバイナリー X.509 delta CRL です。

3. Directory Server にアクセスするために使用する Certificate Manager のバインド DN を設定します。

   Certificate Manager は、証明書と CRL をディレクトリーに公開するために、ディレクトリーへの読み取り/書き込み権限を持っている必要があります。これにより、Certificate Manager は、証明書関連情報を含むユーザーエントリーと、CA の証明書および CRL 関連情報を含む CA エントリーを変更できます。

   バインド DN エントリーは、以下のいずれかになります。

   - Directory Manager などの書き込みアクセスを持つ既存の DN。
   - 書き込みアクセスが付与された新規ユーザー。エントリーは、Certificate Manager の DN で識別することができます (例: **cn=testCA, ou=Research Dept, o=Example Corporation, st=California, c=US**)。



注記

このユーザーに付与される特権を慎重に検討してください。このユーザーは、アカウントの ACL を作成して、そのディレクトリーに書き込みできます。Certificate Manager のエントリーへの書き込みアクセス権限を付与する方法については、Directory Server のドキュメントを参照してください。

4. Certificate Manager が Directory Server に対して認証する方法のディレクトリー認証方法を設定します。Basic 認証 (簡易ユーザー名およびパスワード)、クライアント認証なしの TLS、およびクライアント認証を使用する TLS (証明書ベース) の 3 つのオプションがあります。

   サーバーとの通信方法の設定は、Red Hat Directory Server のドキュメントを参照してください。

7.4.2. LDAP パブリッシャーの設定

Certificate Manager は、LDAP 公開に関連するパブリッシャーのセットを作成、設定、および有効にします。デフォルトのパブリッシャー (CA 証明書、ユーザー証明書、CRL、およびクロスのペア証明書用) は、証明書および CRL を保存するための X.500 標準属性に準拠しており、変更する必要はありません。

表7.1 LDAP パブリッシャー

パブリッシャー	説明
LdapCaCertPublisher	CA 証明書を LDAP ディレクトリーに公開します。
LdapCrlPublisher	CRL を LDAP ディレクトリーに公開します。
LdapDeltaCrlPublisher	デルタ CRL を LDAP ディレクトリーに公開します。
LdapUserCertPublisher	すべての種類のエンドエンティティー証明書を LDAP ディレクトリーに公開します。
LdapCrossCertPairPublisher	相互署名付き証明書を LDAP ディレクトリーに公開します。

7.4.3. マッパーの作成

マッパーは LDAP 公開のみで使用されます。マッパーは、証明書のサブジェクト名と、証明書が公開されるディレクトリーエントリーの DN 間の関係を定義します。Certificate Manager は、使用するエントリーを判断できるように、証明書または証明書要求からエントリーの DN を取得する必要があります。マッパーは、ユーザーエントリーの DN と証明書のサブジェクト名またはその他の入力情報との関係を定義して、エントリーの正確な DN を特定し、ディレクトリーで見つけることができるようにします。

設定すると、Certificate Manager は、最も一般的な関係を定義するマッパーのセットを自動的に作成します。デフォルトのマッパーは、表7.2「デフォルトのマッパー」 に一覧表示されます。

表7.2 デフォルトのマッパー

マッパー	説明
LdapUserCertMap	ユーザー証明書を公開するために、ディレクトリーでユーザーエントリーの正しい属性を見つけます。
LdapCrlMap	CRL を公開するために、ディレクトリーで CA のエントリーの正しい属性を見つけます。
LdapCaCertMap	CA 証明書を公開するために、ディレクトリーで CA のエントリーの正しい属性を見つけます。

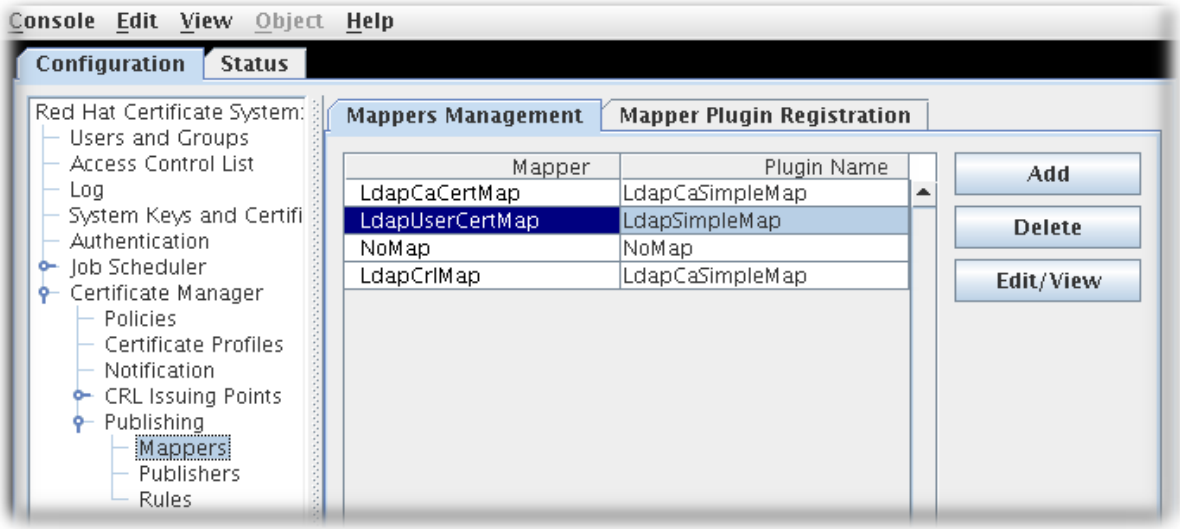
デフォルトのマッパーを使用するには、DN パターンを指定し、ディレクトリーに CA エントリーを作成するかどうかを指定して、各マクロを設定します。他のマッパーを使用するには、マッパーのインスタンスを作成および設定します。詳細は、「マッパープラグインモジュール」を参照してください。

1. Certificate Manager コンソールにログインします。

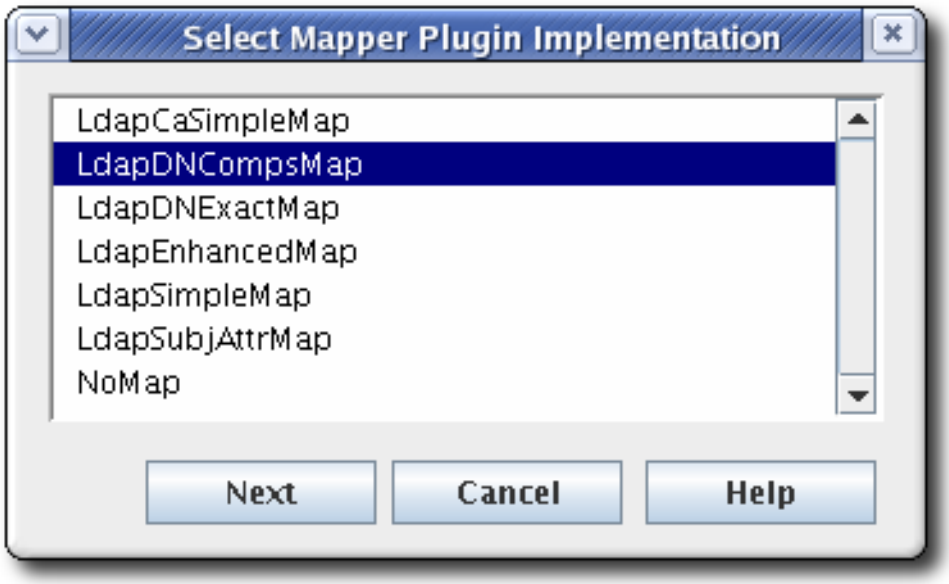
   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Mappers** を選択します。

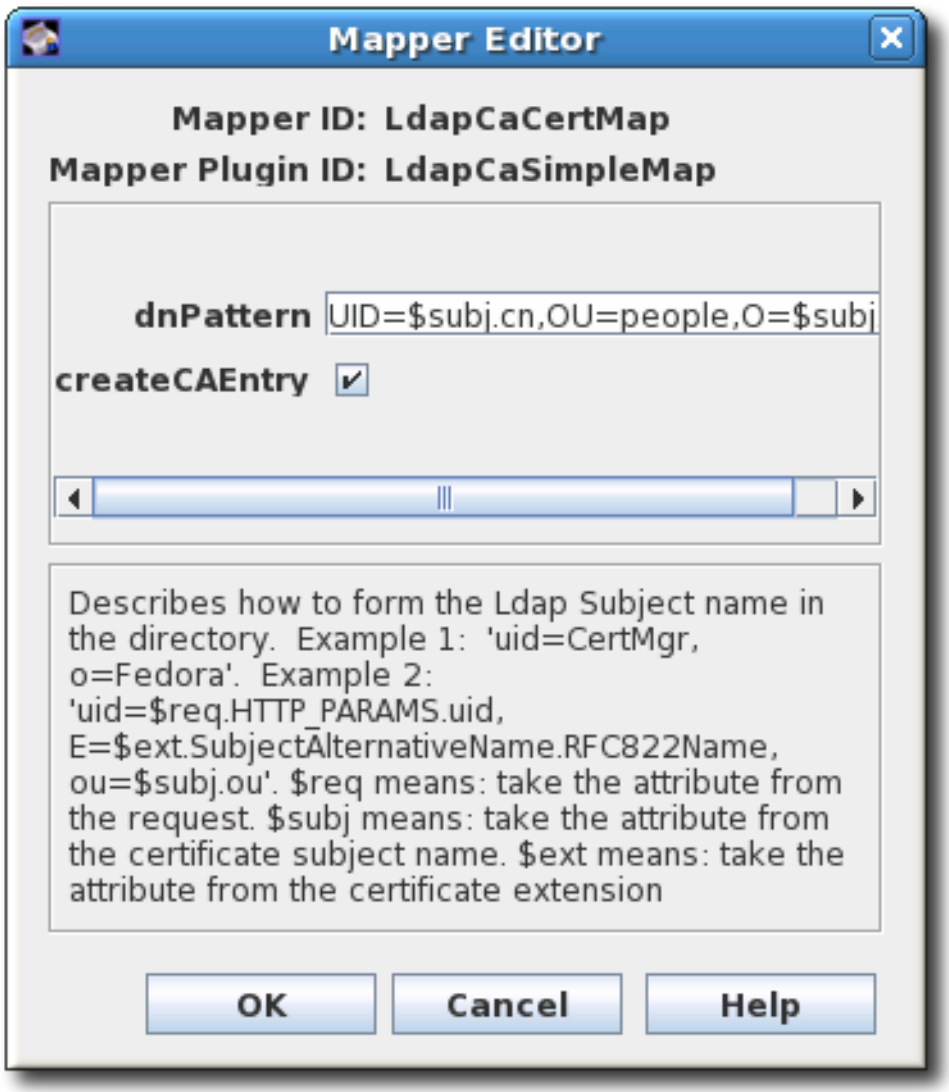
   設定されたマッパーを一覧表示する **Mappers Management** タブが右側で開きます。



3. 新しいマッパーインスタンスを作成するには、**Add** をクリックします。**Select Mapper Plugin Implementation** ウィンドウが開き、登録されたマッパーモジュールが一覧表示されます。モジュールを選択し、そのモジュールを編集します。これらのモジュールに関する詳細は、「マッパープラグインモジュール」を参照してください。



4. マッパーインスタンスを編集し、**OK** をクリックします。



各マッパーに関する詳細は、「マッパープラグインモジュール」 を参照してください。

7.4.4. 設定の完了: ルールおよび有効化

LDAP 公開のマッパーを設定したら、「ルールの作成」 の説明に従って、公開証明書および CRL のルールを設定します。

設定が完了したら、「公開の有効化」 の説明に従って公開を有効にします。

7.5. ルールの作成

ルールは、どの場所にどの証明書オブジェクトを公開するかを決定します。ルールは、連携してではなく、独立して機能します。公開される証明書または CRL は、すべてのルールに対して照合されます。一致するルールはすべてアクティブになります。このようにして、ファイルベースのルール、OCSP ルール、およびディレクトリーベースのルールを照合することにより、同じ証明書または CRL をファイル、Online Certificate Status Manager、および LDAP ディレクトリーに公開できます。

ルールは、各オブジェクトタイプ (CA 証明書、CRL、ユーザー証明書、およびクロスペアの証明書) に設定できます。ルールは、さまざまな種類の証明書またはさまざまな種類の CRL についてより詳細にすることができます。

ルールは最初に、ルールに設定されたタイプと述語をオブジェクトと照合することにより、オブジェクトが一致するかどうかを判断します。一致するオブジェクトは、ルールに関連付けられたパブリッシャーとマッパーにより公開されます。

Certificate Manager が発行する証明書のタイプごとにルールが作成されます。

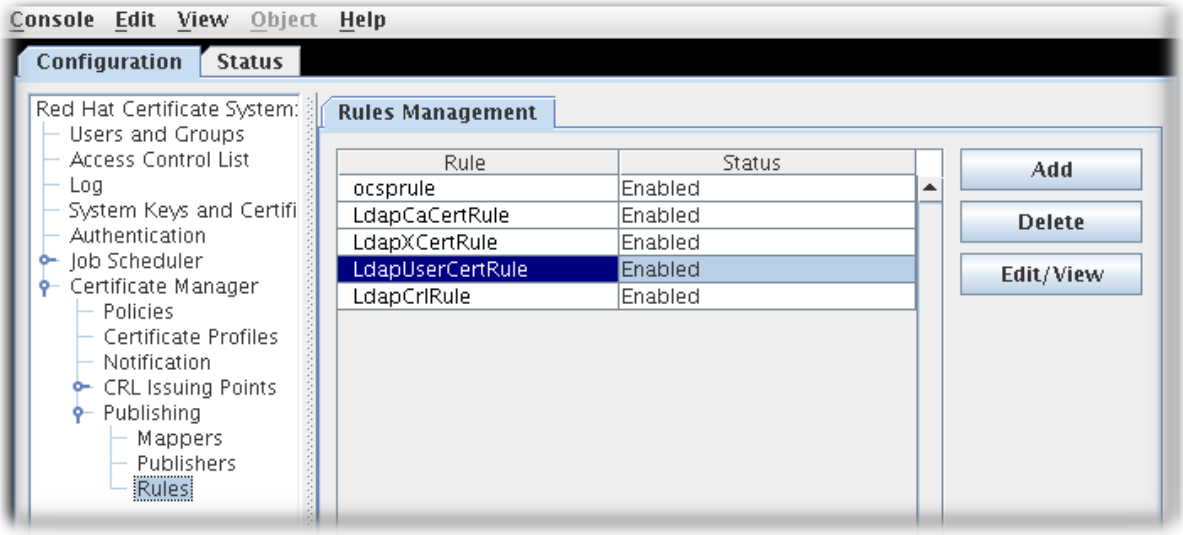
次の手順を実行して、公開ルールを変更します。

1. Certificate Manager コンソールにログインします。

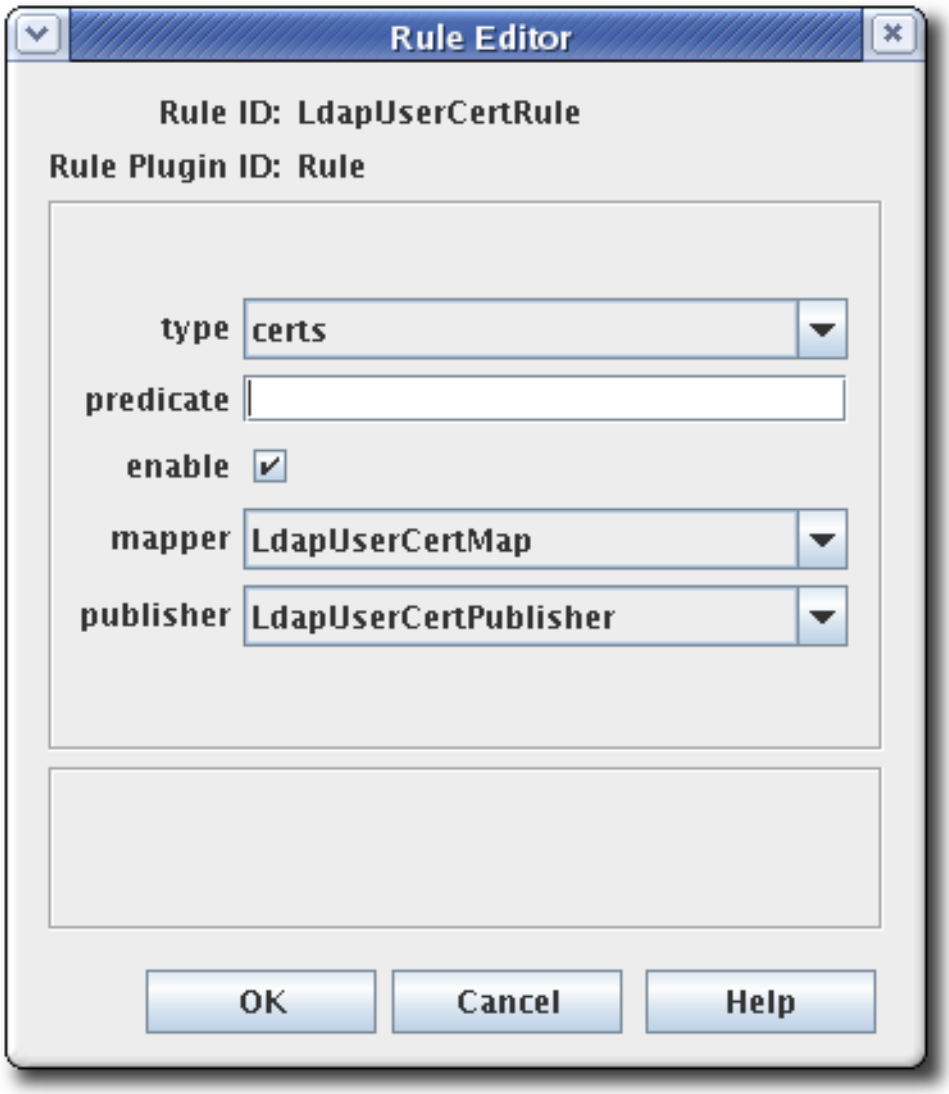
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Rules** を選択します。

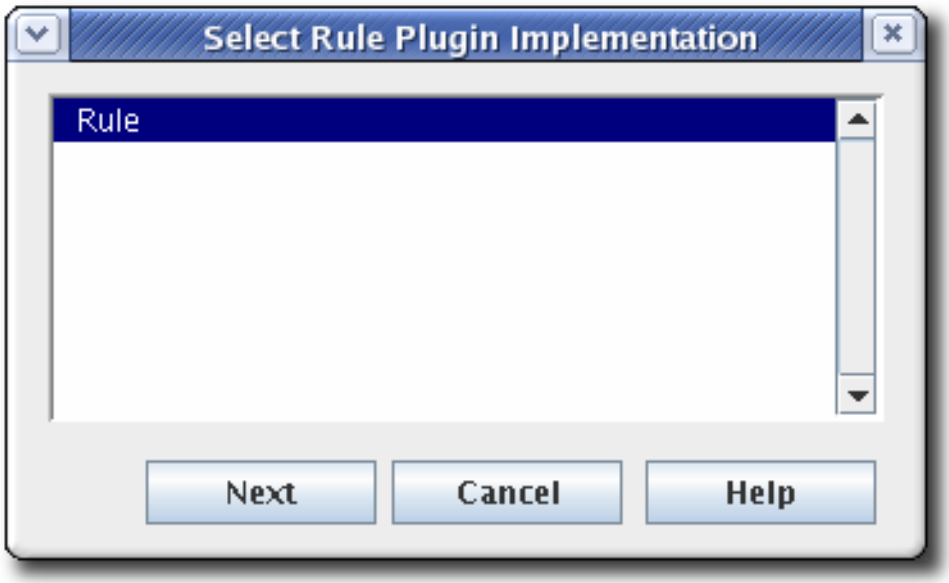
   設定したルールを一覧表示する **Rules Management** タブが右側で開きます。



3. 既存のルールを編集するには、一覧からそのルールを選択して、**Edit** をクリックします。これにより、**Rule Editor** ウィンドウが開きます。

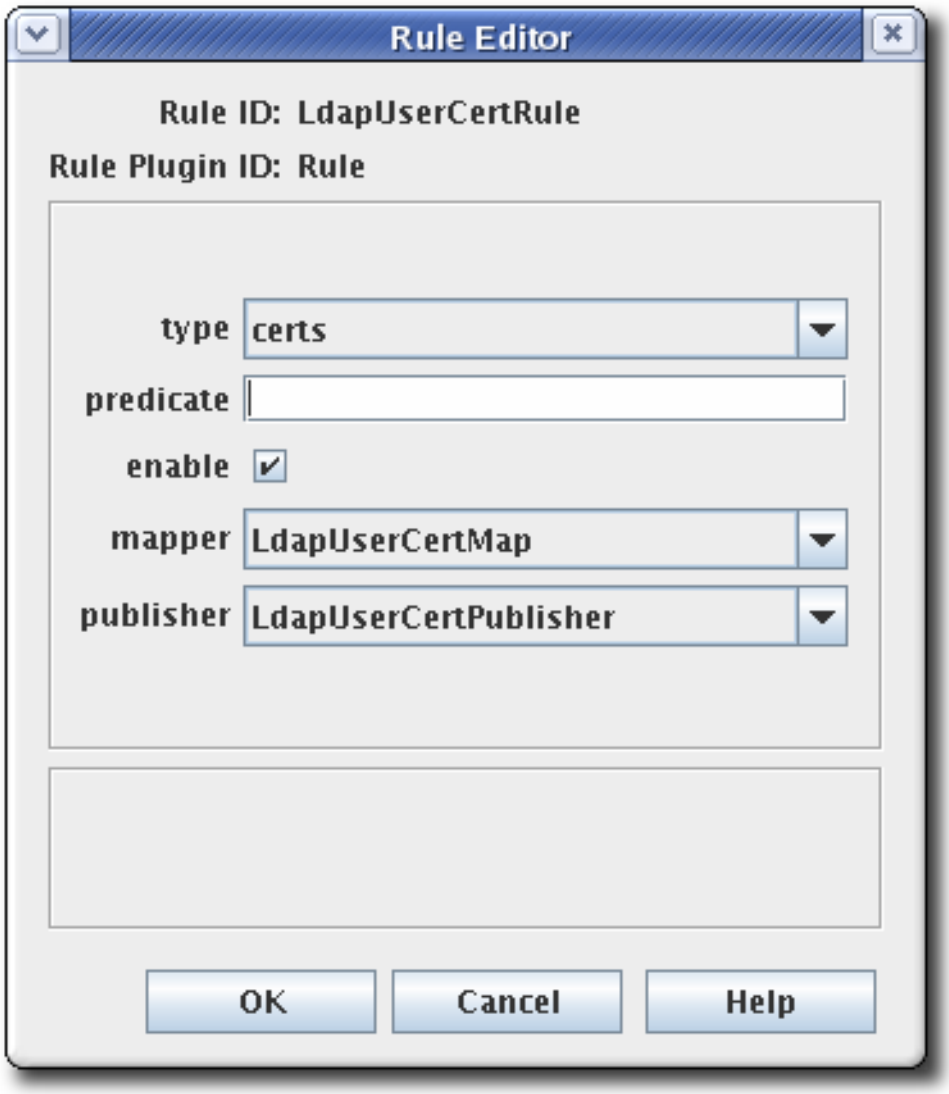


4. ルールを作成するには、**Add** をクリックします。これにより、**Select Rule Plug-in Implementation** ウィンドウが開きます。



Rule モジュールを選択します。これは唯一のデフォルトモジュールです。カスタムモジュールが登録されている場合は、それらも利用できます。

5. ルールを編集します。



- **type**。これは、ルールが適用される証明書のタイプです。CA 署名証明書の場合、値は **cacert** です。自己署名証明書の場合、値は **xcert** です。その他すべてのタイプの証明書の場合、値は **certs** です。CRL には **crl** を指定します。
- **predicate**。これにより、このルールが適用される証明書または CRL 発行ポイントのタイプに述語の値を設定します。CRL 発行ポイント、デルタ CRL、および証明書の述語値の一覧は 表7.3「述語式」 に表示されます。
- **enable**。
- **mapper**。ファイルに公開する場合、マッパーは必要ありません。LDAP 公開にのみ必要です。このルールが LDAP ディレクトリーに公開されるパブリッシャーに関連付けられている場合は、ここに適切なマッパーを選択します。他のすべての形式の発行では空白のままにします。
- **publisher**。ルールに関連付けるパブリッシャーを設定します。

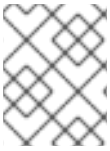
表7.3「述語式」 は、CRL 発行ポイントおよびデルタ CRL および証明書プロファイルを識別するために使用できる述語を一覧表示します。

表7.3 述語式

述語タイプ	述語
CRL 発行ポイント	**issuingPointId==**Issuing_Point_Instance_ID **&& isDeltaCRI==[true|false]** マスター CRL のみを公開するには、**isDeltaCRI==false** を設定します。デルタ CRL のみを公開するには、**isDeltaCRI==true** を設定します。両方を公開するには、マスター CRL のルールとデルタ CRL の別のルールを設定します。
証明書プロファイル	**profileId==**profile_name 使用するプロファイルに基づいて証明書を公開するには、**profileId==** を **caServerCert** などのプロファイル名に設定します。

7.6. 公開の有効化

公開は、ファイル、LDAP、またはその両方に対して有効にできます。パブリッシャー、ルール、およびマッパーの設定後に公開を有効にする必要があります。有効にすると、サーバーは公開を開始しようとします。公開が有効になる前に正しく設定されていなかった場合、公開は望ましくない動作を示したり、失敗したりする可能性があります。



注記

CRL を設定します。CRL は公開前に設定する必要があります。*6章 証明書の取り消しおよび CRL 発行* を参照してください。

1. Certificate Manager コンソールにログインします。

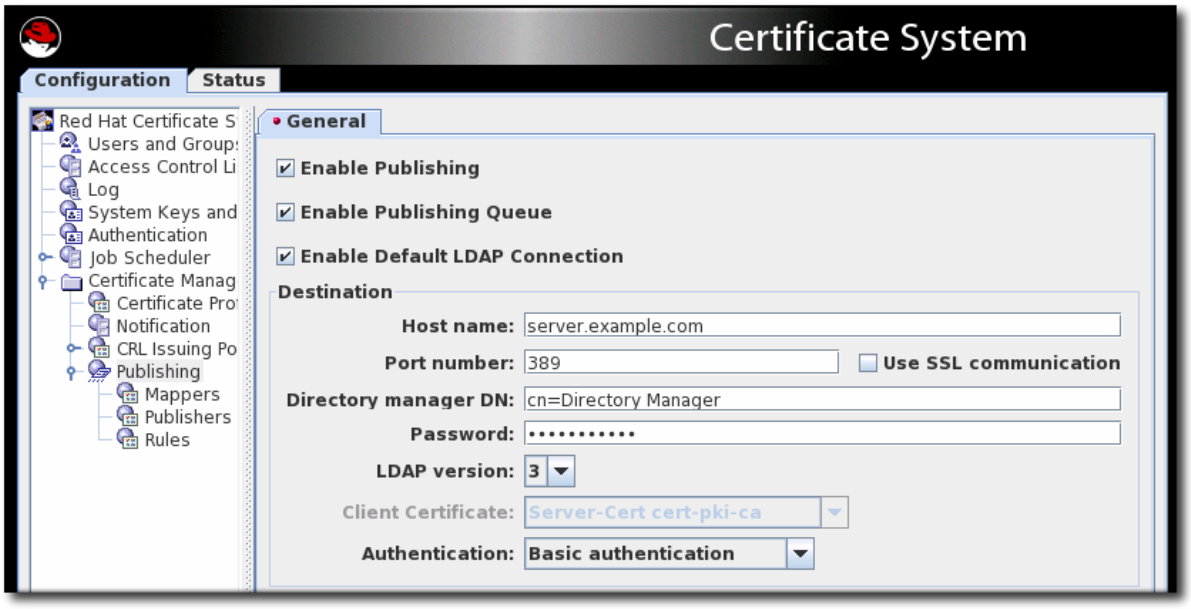
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択します。

   右側のペインには、LDAP 準拠のディレクトリーに公開するための詳細情報が表示されます。

3. ファイルの公開のみを有効にするには、**Enable Publishing** を選択します。

4. LDAP 公開を有効にするには、**Enable Publishing** および **Enable Default LDAP Connection** の両方を選択します。



Destination セクションで、Directory Server インスタンスの情報を設定します。

- **Host name**。Directory Server が TLS クライアント認証通信用に設定されている場合、名前は Directory Server の TLS サーバー証明書のサブジェクト DN の **cn** コンポーネントと一致する必要があります。

  ホスト名は完全修飾ドメイン名または IPv4 アドレスまたは IPv6 アドレスになります。

- **Port number**。

- **Directory Manager DN**これは、Directory Manager 権限を持つディレクトリーエントリーの識別名 (DN) です。Certificate Manager はこの DN を使用してディレクトリーツリーにアクセスし、そのディレクトリーに公開します。この DN に設定されたアクセス制御は、Certificate Manager が公開できるかどうかを判断します。公開システムが実際に書き込む必要のある属性に対してのみ読み取り/書き込み権限が制限されている別の DN を作成することができます。

- **Password**。これは、CA が証明書または CRL の公開先の LDAP ディレクトリーにバインドするために使用するパスワードです。Certificate Manager は、このパスワードを **password.conf** ファイルに保存します。以下に例を示します。

  ```
  CA LDAP Publishing:password
  ```

- **クライアント証明書**。これにより、Certificate Manager が TLS クライアント認証に使用する証明書が公開ディレクトリーに設定されます。デフォルトでは、Certificate Manager は TLS サーバー証明書を使用します。

- **LDAP バージョン**。LDAP バージョン 3 を選択します。

- **Authentication**。Certificate Manager が Directory Server に対して認証する方法。 **Basic authentication** および **TLS client authentication** を選択できます。

  Directory Server が基本認証またはクライアント認証なしの TLS 通信用に設定されている場合は、**Basic authentication** を選択し、Directory Manager DN およびパスワードの値を指定します。

  Directory Server がクライアント認証による TLS 通信用に設定されている場合は、**TLS client authentication** および **Use TLS communication** オプションを選択し、Certificate

Manager がディレクトリーへの TLS クライアント認証に使用する必要のある証明書を特定します。

サーバーは、Directory Server への接続を試みます。情報が正しくない場合、サーバーにはエラーメッセージが表示されます。

7.7. 再開可能な CRL ダウンロードの設定

Certificate System は、中断された CRL ダウンロードをスムーズに再開するオプションを提供します。これは、HTTP 経由でプレーンファイルとして CRL を公開することで行われます。CRL のダウンロード方法により、CRL の取得やネットワーク全体の輻輳を軽減することができます。

7.7.1. wget を使用した CRL の取得

CRL は HTTP 経由でテキストファイルとして公開されるため、**wget** などのツールを使用して CA から手動で取得できます。**wget** コマンドを使用すると、公開されている CRL を検索できます。たとえば、以前のフル CRL よりも新しいフル CRL を取得するには、次のようにします。

```
[root@server ~]# wget --no-check-certificate -d
https://server.example.com:8443/ca/ee/ca/crl/MasterCRL.bin
```

wget の関連パラメーターは 表7.4「CRL の取得に使用する wget オプション」 にまとめています。

表7.4 CRL の取得に使用する wget オプション

引数	説明
引数なし	完全な CRL を取得します。
-N	ローカルコピー (delta CRL) よりも新しい CRL を取得します。
-c	部分的にダウンロードしたファイルを取得します。
--no-check-certificate	接続に TLS をスキップするため、ホストとクライアントの間で TLS を設定する必要はありません。
-d	デバッグ情報を出力します。

7.8. ペア間の証明書の公開

クロスペアの証明書は LDAP ディレクトリーまたはファイルに **crossCertificatePair** エントリーとして公開できます。これはデフォルトで有効になっています。これが無効な場合は、Certificate Manager Console で以下のコマンドを実行して再度有効にできます。

1. CA コンソールを開きます。

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

2. **Configuration** タブで、左側のペインの **Certificate Manager** リンクを選択してから、**Publishing** リンクを選択します。

3. **Publishing** の **Rules** リンクをクリックし ます。これにより、右側に **Rules Management** ペインが開きます。

4. ルールが存在し、無効になっている場合は、**enable** チェックボックスを選択します。ルールが削除された場合は、**Add** クリックして新しいルールを作成します。

   a. **type** ドロップダウンメニューから **xcerts** を選択します。

   b. **有効** のチェックボックスが選択されていることを確認してください。

   c. **mapper** ドロップダウンメニューから、**LdapCaCertMap** を選択します。

   d. **publisher** ドロップダウンメニューから **LdapCrossCertPairPublisher** を選択します。

公開ルールで指定されたマッパーとパブリッシャーは両方とも、CA コンソールの左側のナビゲーションウィンドウの **Publishing** リンクの下の **Mapper** と **Publisher** 下に一覧表示されます。デフォルトでは、マッパーの **LdapCaCertMap** は、**LdapCaSimpleMap LDAP** エントリーに **crossCertificatePair** を保存するように指定します。パブリッシャー **LDAPCrossPairPublisher** はデフォルトで、CA エントリーにクロスペア証明書を **crossCertificatePair;binary** に保存する属性を設定します。

7.9. ファイルへの公開テスト

Certificate Manager が証明書と CRL を正しくファイルに正常に公開していることを確認するには、以下を実行します。

1. CA のエンドエンティティーを開き、証明書をリクエストします。

2. 必要に応じて、エージェントサービスページを使用してリクエストを承認します。

3. エンドエンティティーページから証明書を取得し、証明書をブラウザーにダウンロードします。

4. サーバーが、証明書を含む DER でエンコードされたファイルを生成したかどうかを確認します。

   証明書のバイナリーブロブが公開されることになっているディレクトリーを開きます。証明書ファイルの名前は **cert-serial_number.der** にする必要があります。

5. Binary to ASCII ツールを使用して、DER でエンコードされた証明書をベース 64 でエンコードされた形式に変換します。このツールの詳細は、**BtoA(1)** man ページを参照してください。

   ```
   BtoA input_file output_file
   ```

   input_file は、DER でエンコードされた証明書が含まれるファイルへのパスを設定し、**output_file** は、base-64 でエンコードされた証明書を書き込むようにファイルへのパスを設定します。

6. ASCII ファイルを開きます。base-64 でエンコードされた証明書は以下のように表示されます。

   ```
   -----BEGIN CERTIFICATE-----
   MMIIBtgYJYIZIAYb4QgIFoIIBpzCCAZ8wggGbMIIBRaADAgEAAgEBMA0GCSqGSIb3DQEBB
   AUAMFcxC
   AJBgNVBAYTAlVTMSwwKgYDVQQKEyNOZXRzY2FwZSBDb21tdW5pY2F0aWhfyyuougjgjjg
   mkgjkgmjg
   fjfgjjjgfyjfyj9ucyBDb3Jwb3JhdGlvbjpMEaMBgGA1UECxMRSXNzdWluZyhgdfhbfdpffjphotoo
   ```

```
gdhkBBdXRob3JpdHkwHhcNOTYxMTA4MDkwNzM0WhcNOTgxMTA4MDkwNzMM0WjBXMQ
swCQYDVQQGEwJ
VUzEsMCoGA1UEChMjTmV0c2NhcGUgQ29tbXVuaWNhdGlvbnMgQ29ycG9yY2F0aW9ucyB
Db3Jwb3Jhd
GlvbjpMEaMBgGA1UECxMRSXNzdWluZyBBdXRob3JpdHkwHh
-----END CERTIFICATE-----
```

7. Pretty Print Certificate ツールを使用して、ベース 64 でエンコードされた証明書を読み取り可能なフォームに変換します。このツールの詳細は、**PrettyPrintCert(1)** man ページを参照してください。

```
PrettyPrintCert input_file [output_file]
```

input_file は base-64 でエンコードされた証明書が含まれる ASCII ファイルへのパスを設定し、任意で **output_file** に証明書を書き込むファイルにパスを設定できます。出力ファイルが設定されていない場合、証明書情報は標準出力に書き込まれます。

8. 出力と、発行された証明書を比較します。証明書のシリアル番号と、ファイル名で使用されている証明書と比較します。

   すべてが一致する場合、Certificate Manager は証明書をファイルに公開するように正しく設定されています。

9. 証明書を取り消します。

10. サーバーが、CRL を含む DER でエンコードされたファイルを生成したかどうかを確認します。

    サーバーが CRL をバイナリーブロブとして公開するディレクトリーを開きます。CRL ファイルの名前は **crl-this_update.der** であるはずです。**this_update** は、CRL の時間依存の **This Update** 変数から派生した値を指定します。

11. Binary to ASCII ツールを使用して、DER でエンコードされた CRL をベース 64 でエンコードされた形式に変換します。

```
BtoA input_file output_file
```

12. Pretty Print CRL ツールを使用して、base 64 でエンコードされた CRL を読み取り可能なフォームに変換します。

```
PrettyPrintCrl input_file [output_file]
```

13. 出力を比較します。

7.10. ファイルに公開される証明書および CRL の表示

証明書と CRL は、base-64 でエンコードされたファイルまたは DER エンコードの 2 種類のファイルに公開できます。これらのファイルの内容は、**dumpasn1** ツールまたは **PrettyPrintCert** または **PrettyPrintCrl** ツールを使って Pretty-print 形式にこのファイルを変換して表示できます。

base-64 でエンコードされたファイルのコンテンツを表示するには、以下を実行します。

1. base-64 ファイルをバイナリーに変換します。以下に例を示します。

```
AtoB /tmp/example.b64 /tmp/example.bin
```

2. **PrettyPrintCert** または **PrettyPrintCrl** ツールを使用して、バイナリーファイルを pretty-print 形式に変換します。以下に例を示します。

```
PrettyPrintCert example.bin example.cert
```

DER でエンコードされたファイルの内容を表示するには、DER エンコードファイルで **dumpasn1**、**PrettyPrintCert**、または **PrettyPrintCrl** ツールを実行するだけです。以下に例を示します。

```
PrettyPrintCrl example.der example.crl
```

7.11. ディレクトリーの証明書および CRL の更新

Directory Server がダウンしているときに証明書が発行または取り消されると、Certificate Manager と公開ディレクトリーが同期しなくなる可能性があります。発行または失効した証明書は、Directory Server のバックアップ時に手動で公開または公開解除する必要があります。

ディレクトリーと同期していない証明書 (ディレクトリーにない有効な証明書と、ディレクトリーに残っている失効または期限切れの証明書) を見つけるために、Certificate Manager は、内部データベース内の証明書がディレクトリーに公開されているかどうかの記録を保持します。Certificate Manager および公開ディレクトリーの同期がなくなる場合は、Certificate Manager エージェントサービスページの **Update Directory** オプションを使用して、公開ディレクトリーを内部データベースと同期します。

ディレクトリーと内部データベースの同期には、以下の選択肢を利用できます。

- 内部データベースで同期していない証明書を検索し、公開または非公開にします。
- Directory Server のダウン中に発行された証明書を公開します。同様に、Directory Server の停止時に取り消された、または有効期限が切れた証明書も使用できます。
- シリアル番号 **xx** からシリアル番号 **yy** までのシリアル番号に基づいて、一連の証明書を公開または非公開にします。

Certificate Manager の公開ディレクトリーは、Certificate Manager エージェントからのみ手動で更新できます。

7.11.1. ディレクトリーでの証明書の手動による更新

Certificate Manager エージェントサービスページの **Update Directory Server** フォームを使用すると、証明書関連の情報を使用してディレクトリーを手動で更新できます。このフォームは、以下の操作の組み合わせを開始します。

- 証明書でディレクトリーを更新します。
- 期限切れの証明書をディレクトリーから削除します。

  自動化されたジョブをスケジュールすることにより、公開ディレクトリーからの期限切れの証明書の削除を自動化できます。
- ディレクトリーから失効した証明書を削除します。

以下のコマンドを実行して、ディレクトリーを変更で手動で更新します。

1. Certificate Manager エージェントサービスページを開きます。
2. **Update Directory Server** のリンクを選択します。
3. 適切なオプションを選択し、**Update Directory** をクリックします。

   Certificate Manager は、内部データベースの証明書情報でディレクトリーの更新を開始します。大幅に変更した場合は、ディレクトリーの更新にかなりの時間がかかる可能性があります。この期間中、発行された証明書や取り消された証明書など、Certificate Manager を介して行われた変更は、更新に含まれない場合があります。ディレクトリーの更新中に証明書が発行または取り消された場合は、それらの変更を反映するようにディレクトリーを再度更新してください。

ディレクトリーの更新が完了すると、Certificate Manager にステータスレポートが表示されます。プロセスが中断された場合、サーバーはエラーメッセージを記録します。

Certificate Manager がルート CA としてインストールされている場合、エージェントインターフェイスを使用してディレクトリーを有効な証明書で更新するときに、ユーザー証明書用に設定された公開ルールを使用して CA 署名証明書が公開されることがあります。これにより、オブジェクトクラスの違反エラーや他のマッパーの他のエラーが返される場合があります。CA 署名証明書を除外するために適切なシリアル番号範囲を選択すると、この問題を回避できます。CA 署名証明書は、ルート CA の最初の証明書です。

- **predicate** パラメーターの値を **profileId!=caCACert** に変更して、ユーザー証明書のデフォルト公開ルールを変更します。
- **LdapCaCertPublisher** プラグインモジュールを使用して別のルールを追加し、predicate パラメーターを **profileId=caCACert** に設定して CA 証明書を公開します。

7.11.2. ディレクトリーでの CRL の手動による更新

Certificate Manager エージェントサービスページの **Certificate Revocation List** フォームは、CRL 関連の情報のあるディレクトリーを手動で更新します。

以下のコマンドを実行して CRL 情報を手動で更新します。

1. Certificate Manager エージェントサービスページを開きます。
2. **Update Revocation List** を選択します。
3. **Update** をクリックします。

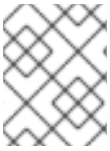
Certificate Manager は、内部データベースの CRL でディレクトリーの更新を開始します。CRL のサイズが大きい場合は、ディレクトリーの更新にかなり時間がかかります。この期間中、CRL への変更は更新に含まれない可能性があります。

ディレクトリーを更新すると、Certificate Manager にステータスレポートが表示されます。プロセスが中断された場合、サーバーはエラーメッセージを記録します。

第8章 証明書を登録するための認証

8.1. 認証プラグインによる自動承認

プロファイルの ***auth.instance_id*** パラメーターは認証メカニズムを指定します。証明書要求は、認証プラグインで自動的に承認されるか、CA エージェントによって手動で承認されます。



注記

証明書登録プロファイルの編集方法は、「証明書プロファイルの設定」 を参照してください。

8.1.1. 登録要求の自動承認の設定

登録リクエストは自動的に承認される設定は、要求の種類によって異なります。

- agent-pre-approved CMC 要求の場合は、CA プロファイルに設定されます。

  ```
  auth.instance_id=CMCAuth
  authz.acl=group="Certificate Manager Agents"
  ```

  authz.acl パラメーターは、リクエストを承認できるグループを定義します。

- ユーザーが開始された要求の場合:
  - CMC Shared Token を使用する場合は、CA プロファイルで設定します。

    ```
    auth.instance_id=CMCUserSignedAuth
    ```

    必要なデフォルトと制約:

    ```
    policyset.cmcUserCertSet.1.constraint.class_id=cmcSharedTokenSubjectNameConstraintImpl
    policyset.cmcUserCertSet.1.constraint.name=CMC Shared Token Subject Name Constraint
    policyset.cmcUserCertSet.1.default.class_id=authTokenSubjectNameDefaultImpl
    policyset.cmcUserCertSet.1.default.name=Subject Name Default
    ```

  - ユーザー署名要求を使用する場合は、CA プロファイルで設定します。

    ```
    auth.instance_id=CMCUserSignedAuth
    ```

    必要なデフォルトと制約:

    ```
    policyset.cmcUserCertSet.1.default.params.name=
    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
    ```

プロファイルの編集に関する詳細は、「証明書プロファイルの設定」 を参照してください。

8.1.2. CMC 認証プラグイン

Certificate System は、以下の認証プラグインを提供します。

CMCAuth

CA エージェントが CMC 要求に署名する場合は、このプラグインを使用します。

CMCAuth プラグインを使用するには、登録プロファイルで以下を設定します。

```
auth.instance_id=CMCAuth
```

デフォルトでは、以下の登録プロファイルは **CMCAuth** プラグインを使用します。

- システム証明書の場合:
  - **caCMCauditSigningCert**
  - **caCMCcaCert**
  - **caCMCECserverCert**
  - **caCMCECsubsystemCert**
  - **caCMCECUserCert**
  - **caCMCkraStorageCert**
  - **caCMCkraTransportCert**
  - **caCMCocspCert**
  - **caCMCserverCert**
  - **caCMCsubsystemCert**
- ユーザー証明書の場合:
  - **caCMCUserCert**
  - **caECFullCMCUserCert**
  - **caFullCMCUserCert**

CMCUserSignedAuth

署名付きまたは SharedSecret ベースの CMC 要求を送信する場合は、このプラグインを使用します。

CMCUserSignedAuth プラグインを使用するには、登録プロファイルに以下を設定します。

```
auth.instance_id=CMCUserSignedAuth
```

ユーザー署名の CMC 要求は、要求された証明書と同じ **subjectDN** 属性が含まれるユーザーの証明書で署名する必要があります。ユーザーが署名した CMC 要求を使用できるのは、ユーザーが他の証明書のユーザー ID を証明するために使用できる署名証明書を既に取得している場合のみです。

SharedSecret ベースの CMC 要求は、要求が要求自体の秘密鍵によって署名されたことを意味しま

す。この場合、CMC 要求は認証に共有秘密メカニズムを使用する必要があります。SharedSecret ベースの CMC 要求は通常、ユーザーの最初の署名証明書を取得するために使用され、後で他の証明書を取得するために使用されます。詳細は、「CMC SharedSecret 認証」を参照してください。

デフォルトでは、以下の登録プロファイルは、**CMCUserSignedAuth** プラグインを使用します。

- **caFullCMCUserSignedCert**
- **caECFullCMCUserSignedCert**
- **caFullCMCSharedTokenCert**
- **caECFullCMCSharedTokenCert**

8.1.3. CMC SharedSecret 認証

Shared Secret 機能を使用して、ユーザーがサーバーに署名されていないリクエストを送信できるようにします。たとえば、ユーザーが最初の署名証明書を取得する場合は、これが必要になります。この署名証明書は、後でこのユーザーの他の証明書に署名するために使用できます。

8.1.3.1. 共有シークレットトークンの作成

『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』の『The Shared Secret Workflow』セクションでは、Shared Secret Token を使用する場合のワークフローについて説明しています。状態に応じて、エンドエンティティーユーザーまたは管理者が共有シークレットトークンを作成します。

注記

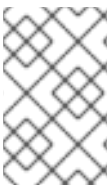
共有シークレットトークンを使用するには、Certificate System で RSA 発行の保護証明書を使用する必要があります。詳細は、RHCS P 計画、インストール、およびデプロイメントのガイドの共有シークレット機能の有効化セクションを参照してください。

Shared Secret Token を作成するには、以下を入力します。

```
# 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"
```

HSM を使用する場合は、さらに HSM セキュリティートークン名を設定するコマンドの **-h *token_name*** オプションを渡します。

CMCSharedToken ユーティリティーの詳細は、CMCSharedToken(8) の man ページを参照してください。



注記

生成されたトークンは暗号化され、パスワードを認識したユーザーのみになります。CA 管理者がユーザーのトークンを生成する場合、管理者はセキュアな方法でユーザーにパスワードを提供する必要があります。

Shared Token を作成したら、管理者はトークンをユーザーまたは証明書レコードに追加する必要があります。詳細は、「CMC 共有シークレットの設定」を参照してください。

8.1.3.2. CMC 共有シークレットの設定

管理者は、計画されるアクションに応じて、ユーザーまたは証明書の LDAP エントリーに生成した後に Shared Secret Token を保存する必要があります。

ワークフローの詳細と、いつ共有シークレットを使用するかについては、『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『The Shared Secret Workflow』 セクションを参照してください。

8.1.3.2.1. 証明書の登録用ユーザーエントリーへの CMC 共有シークレットの追加

証明書の登録に Shared Secret Token を使用するには、ユーザーの LDAP エントリーに管理者として保存します。

```
# 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
```

8.1.3.2.2. 証明書失効用の証明書への CMC 共有シークレットの追加

証明書失効に Shared Secret Token を使用するには、取り消される証明書の LDAP エントリーに管理者として保存します。

```
 # 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
```

8.2. CA エージェントによる手動承認

CA プロファイルで手動承認が設定されている場合、ユーザーは CMC 要求を作成できますが、要求は CA エージェントによって手動で承認される必要があります。

CA プロファイルでエージェントの手動承認を設定するには、次の手順を実行します。

1. ***auth.instance_id*** を空の値に設定します。

   ```
   auth.instance_id=
   ```

2. ***authz.acl*** パラメーターを設定しないでください。

プロファイルの編集に関する詳細は、「証明書プロファイルの設定」 を参照してください。

8.3. コマンドラインを使用した証明書ステータスの手動確認

証明書要求を確認するには、証明書要求を承認する適切なパーミッションを持つエージェントとして認証されていることを確認してください。**pki** コマンドラインインターフェイスの設定に関する詳細は、「pki CLI の初期化」 を参照してください。

要求を確認するには、以下を実行します。

1. 保留中の証明書要求の一覧を表示します。

```
$ pki agent_authentication_parameters ca-cert-request-find --status pending
```

このコマンドは、保留中の証明書要求をすべて表示します。

2. 特定の証明書要求をダウンロードします。

```
$ pki agent_authentication_parameters ca-cert-request-review id --file request.xml
```

3. エディターまたは別のターミナルでエディターまたは別のターミナルで ***request.xml*** ファイルを開いて要求の内容を確認して、それが正当であることを確認します。その後、プロンプトに回答します。リクエストが有効な場合は、**approve** と回答して、**Enter** を押します。リクエストが無効の場合は **reject** と回答し、**Enter** を押します。組織は、セマンティックの相違点をサブスクライブして **reject** および **cancel** にすることができます。どちらも証明書は発行されません。

8.4. WEB インターフェイスを使用した証明書ステータスの手動による確認

1. Web ブラウザーで、以下の URL を開きます。

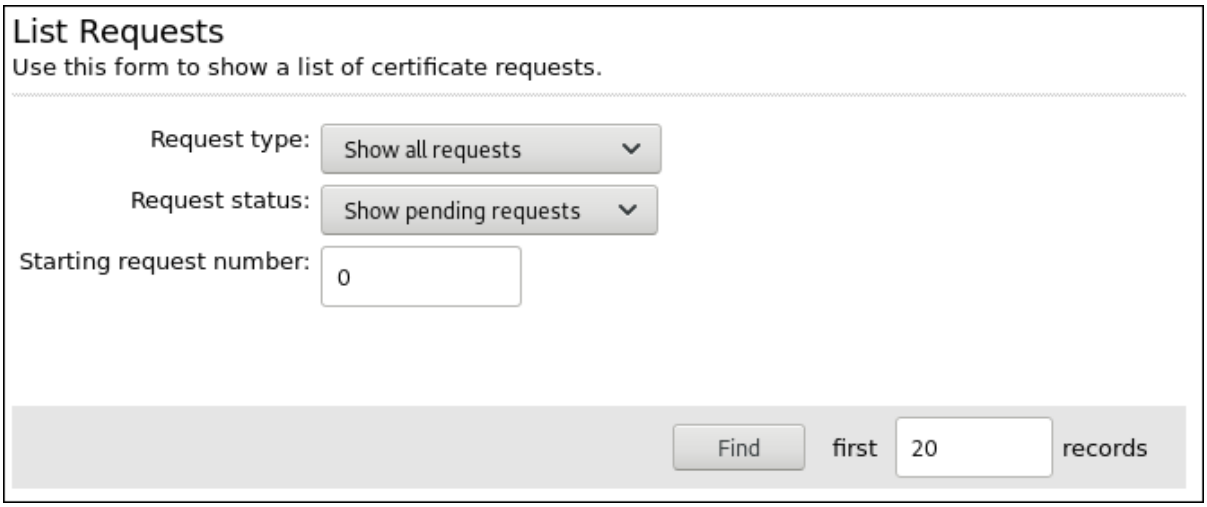
```
https://server_host_name:8443/ca/agent/ca
```

2. エージェントとして認証します。ユーザーとして認証し、ブラウザーの設定に関する詳細は、「ブラウザーの初期化」 を参照してください。

3. 左側のサイドバーの **List requests** リンクをクリックします。

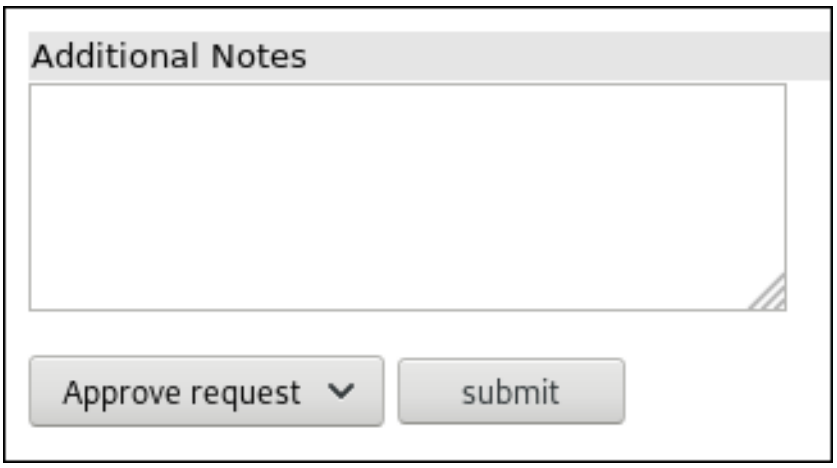
4. **Request type** の **Show all requests** を選択し、**Request status** の Pending requests を選択して**要求** をフィルターリングします。

5. 右下隅にある **Find** をクリックします。



6. 結果ページでは、確認を待機中の保留中のリクエストをすべて表示します。要求番号をクリックして、リクエストを確認します。

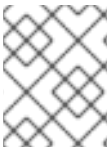
7. 要求情報を確認し、それが正当な要求であることを確認します。必要に応じて、ポリシー情報を変更して間違いを修正したり、証明書に必要な変更 (**not valid after** フィールドなど) を行ったりします。必要に応じて、追加の注記を残しておきます。



ドロップダウンメニューには、複数のレビューステータスの更新が含まれます。**Approve request** を選択してリクエストを承認するか、または **Reject request** を選択して否定してから、**Submit** をクリックします。組織は **Reject request** と **Cancel Request** とのセマンティックの相違点をサブスクライブできます。いずれの場合でも、証明書は発行されません。

第9章 証明書の登録の認可 (アクセス評価者)

本章では、アクセスエバリュエーターを使用した承認メカニズムを説明します。



注記

証明書登録プロファイルの編集方法は、「証明書プロファイルの設定」 を参照してください。

9.1. 承認メカニズム

認証 メカニズムの他に、各登録プロファイルに独自の **承認** メカニズムがあるように設定できます。承認メカニズムは、認証が成功しないと実行されません。

承認メカニズムは、Access Evaluator プラグインフレームワークによって提供されます。アクセスエバリュエーターは、アクセス制御命令 (ACI) エントリーの評価に使用されるプラグ可能なクラスです。このメカニズムは、事前に定義された引数のリスト (つまり **type**、**op**、**value**) などを取り、**group='Certificate Manager Agents'** などの評価を評価し、評価の結果に応じてブール値を返す **評価** 方法を提供します。

9.2. デフォルトの評価者

Red Hat Certificate System は、デフォルトのエバリュエーターを 4 つ提供します。

```
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
```

group アクセスエバリュエーターは、ユーザーのグループメンバーシッププロパティーを評価します。たとえば、以下の登録プロファイルエントリーでは、CA エージェントのみがそのプロファイルで登録できます。

```
authz.acl=group="Certificate Manager Agents"
```

ipaddress アクセスエバリュエーターは、要求側の IP アドレスを評価します。たとえば、以下の登録プロファイルエントリーでは、指定した IP アドレスを持つホストのみがそのプロファイルで登録を行えます。

```
authz.acl=ipaddress="a.b.c.d.e.f"
```

user アクセスエバリュエーターは、完全一致についてユーザー ID を評価します。たとえば、以下の登録プロファイルエントリーでは、リストされたユーザーと一致するユーザーのみが、そのプロファイルを使用した登録を行うことができます。

```
authz.acl=user="bob"
```

user_origreq アクセスエバリュエーターは、認証されたユーザーを、以前に一致した同等の要求に対して評価します。この特別なエバリュエーターは、更新を要求するユーザーが元の要求を所有するユーザーと同じであることを確認するために、更新を目的として特別に設計されています。たとえば、次の更新登録プロファイルエントリーでは、認証されたユーザーの UID は、更新を要求しているユーザーの UID と一致する必要があります。

```
authz.acl=user_origreq="auth_token.uid"
```

新しいエバリュエーターは現在のフレームワークで記述でき、CS コンソールから登録できます。デフォルトのエバリュエーターはテンプレートとして使用して、より多くのターゲットプラグインを拡張し、カスタマイズできます。

パート IV. サブシステムインスタンスの管理

第10章 セルフテスト

10.1. セルフテストの実行

Certificate System には、サーバーのセルフテストを可能にする機能が追加されました。セルフテストは起動時に実行され、オンデマンドで実行することもできます。起動セルフテストはサーバーの起動時に実行され、重要なセルフテストが失敗した場合にサーバーが起動しないようにします。オンデマンドのセルフテストは、サブシステムコンソールのセルフテストボタンをクリックして実行されます。

10.1.1. セルフテストの実行

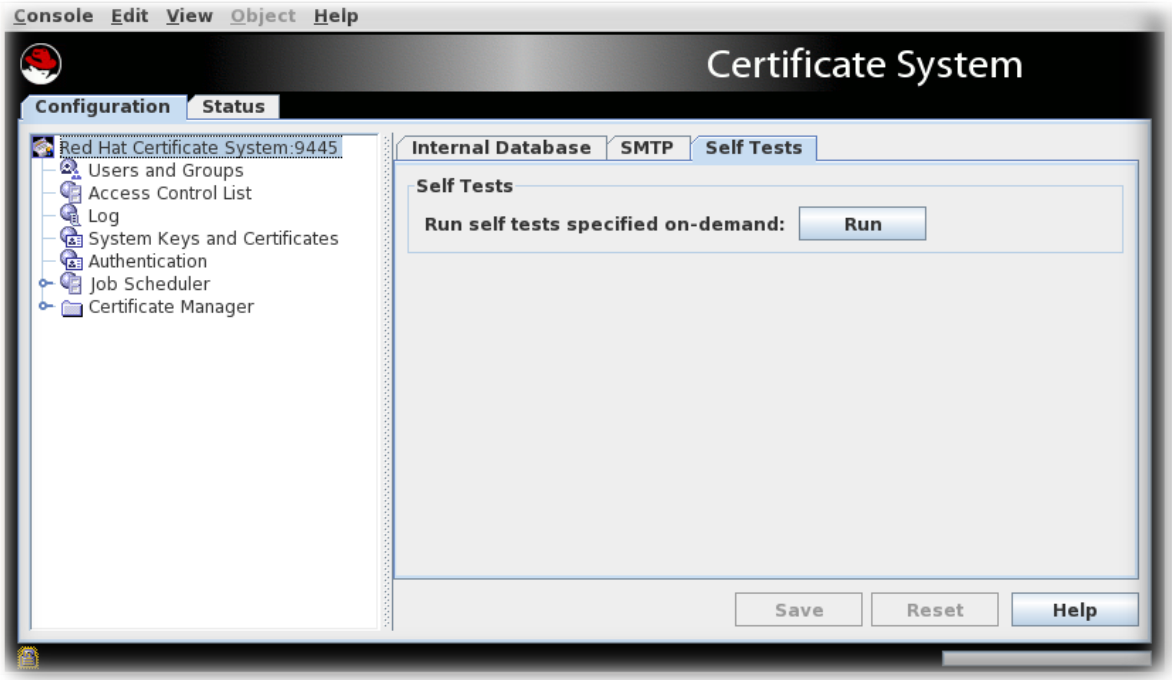
CA、OCSP、KRA、または TKS サブシステムのオンデマンドのセルフテストは、コンソールから実行します。TPS システムのオンデマンドのセルフテストは、Web サービスページから実行されます。

10.1.1.1. コンソールからのセルフテストの実行

1. コンソールにログインします。

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

2. 左側のペインの上部にあるサブシステム名を選択します。



3. **Self Tests** タブを選択します。

4. **Run** をクリックします。

   サブシステムに設定されたセルフテストが実行されます。重大なセルフテストに失敗すると、サーバーが停止します。

5. **On-Demand Self Tests Results** ウインドウが表示され、セルフテストの実行にログイベントが表示されます。

10.1.1.2. TPS セルフテストの実行

コマンドラインインターフェイス (CLI) から TPS のセルフテストを実行するには、以下を実行します。

- **pki tps-selftest-find**
- **pki tps-selftest-run**
- **pki tps-selftest-show**

10.2. セルフテストの失敗のデバッグ

セルフテストが失敗した場合、Certificate System インスタンスは完全に停止し、HTTP または HTTPS 要求に応答しなくなります。

手動で実行されたセルフテストの失敗を診断するには、「セルフテストロギング」 で説明されているさまざまなログを参照してください。多くの場合、デバッグログなど、他のログも役に立ちます。サブシステムログの詳細については、13章 *サブシステムログの設定* を参照してください。デバッグログの詳細については、『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Certificate System Architecture Overview』 にある 『Logs』 セクションを参照してください。

セルフテストの失敗の一般的な原因は、サービス (LDAP など) がダウンしているか到達不能である、証明書が期限切れである、またはシステム設定が間違っていることです。セルフテストの失敗の正確な原因がログに記録されています。

セルフテストの失敗の原因を特定して修正したら、Certificate System サーバーを再起動して通常の操作を再開してください。

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

10.2.1. セルフテストロギング

別のログ(**selftests.log**)が、起動用セルフテストとオンデマンドセルフテストの両方のレポートが含まれるログディレクトリーに追加されます。

第11章 証明書/キー暗号トークンの管理

本章では、さまざまなシナリオの証明書のインポートおよび検証方法など、暗号化トークンで証明書/鍵のデータベースを管理する方法を説明します。

暗号トークンについて
NSS ソフトトークンについては、2.3.8.1 を参照してください。Red Hat Certificate System の計画、インストール、およびデプロイメントガイドの NSS ソフトトークン (内部トークン)。

11.1. CERTUTIL および PKICERTIMPORT

certutil コマンドは、Network Security Services (NSS)によって提供されます。 **certutil** は、証明書の検証およびインポートに使用されます。ただし、**certutil** の使用方法に関する基本的な概要を以下に示します。ただし、**PKICertImport** は、証明書を安全に検証およびインポートするために選択できるラッパースクリプトです。これには、**certutil** を使用して複数のコマンドの呼び出しが必要で、正しい使用方法は本書の範囲外です。

11.1.1. certutil の基本的な使用方法

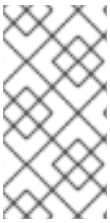
certutil [command] [options]

各 **certutil** 呼び出しは、通常は大文字で表されるコマンドフラグと、コマンドの動作を制御する一連のオプションを取ります。オプションが値を取る場合、その値の名前は<と>間に付けられます。

11.1.2. PKICertImport の基本的な使用方法

PKICertImport [options]

各 **PKICertImport** 呼び出しは、指定された証明書を検証およびインポートする一連のオプションを受け入れます。**certutil** の幅広いユースケースとは異なり、**PKICertImport** は、証明書を安全にインポートおよび検証することのみに重点を置いています。利用可能なオプションの詳細は、「一般的な **certutil** および **PKICertImport** オプション」を参照してください。



注記

PKICertImport は、実行中に NSS DB や HSM のパスワードを複数回要求します。これは、**PKICertImport** と NSS DB との対話を複数回行う必要があるためです。NSS DB パスワードを繰り返し入力しなくてもよいように、**-f <filename> でパスワードファイルを指定します**。完了したら、パスワードファイルを必ず削除してください。

11.1.3. certutil の一般的なコマンド

以下のコマンドは **certutil** に固有のもので、いくつかの一般的なコマンドの概要を説明します。**PKICertImport** は、これらのコマンドフラグと互換性がなく、これらのコマンドフラグも必要ありません。

certutil -A
-A コマンドは、証明書の追加を示しています。インポートする証明書(**-i**)、その証明書のニックネーム(**-n**)、および証明書の一連の信頼フラグ(**-t**)が必要です。

certutil -V
-V コマンドは、証明書の検証を示しています。検証には証明書のニックネーム(**-n**)と、実行する検証のタイプ(**-u**)が必要です。

certutil -D
-D コマンドは、証明書の削除を示しています。削除する証明書のニックネーム(**-n**)が必要です。

証明書の公開鍵部分のみが削除され、秘密鍵が存在する場合は削除されないことに注意してください。

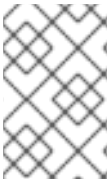
certutil -M
-M コマンドは、証明書の変更を示しています。変更する証明書のニックネーム(**-n**)と、証明書を提供する一連の信頼フラグ(**-t**)が必要です。

certutil -L
-L コマンドは、証明書またはすべての証明書の一覧表示を示しています。ニックネームオプション(**-n**)を指定すると、その証明書に関する詳細情報が一覧表示されます。それ以外の場合は、存在するすべての証明書に関する一般的な情報が一覧表示されます。

certutil -L の結果として、各証明書のニックネームとその信頼情報が表示されます。以下に例を示します。

表11.1 証明書のニックネームと信頼情報

証明書のニックネーム	Trust Attributes SSL, S/MIME, JAR/XPI
caSigningCert pki-ca1	CT、C、C



注記

certutil -L で表示される信頼属性は、**-t** オプションで指定したものに対応します。

certutil -L はデータベースを変更しないため、必要な回数だけ安全に実行できます。

11.1.4. 一般的な certutil および PKICertImport オプション

以下の手順に従って、値は特定のデプロイメントシナリオに適し、正しい値であることを確認します。これらのオプションの多くは、**PKICertImport** でも利用できます。

-n <nickname>
-n <nickname> オプションは、 証明書のニックネームを指定します。これは任意のテキストを指定でき、証明書への参照としてのみ使用されます。一意である必要があります。

設定に応じて、この値を更新してください。

-d <directory>
-d <directory> オプションは、 使用中の NSS DB ディレクトリーへのパスを指定します。通常、このディレクトリーはすでに存在し、.を使用して現在のディレクトリーを参照します。

設定に応じて、この値を更新してください。

-t <trust>
-t <trust> オプションは、 証明書の信頼レベルを指定します。

信頼には、以下の 3 つのカテゴリーがあります。

- TLS の信頼

- メールの信頼
- オブジェクト署名の信頼

各信頼の位置には、信頼のレベルを指定する信頼文字を 1 つまたは複数追加できます。以下の信頼文字は、**c**、**C**、および **T** です。

- **C** は、この証明書が認証局(CA)である必要があります。
- **C** は、サーバー証明書に署名するための信頼できる認証局であることを示しています(**C** は小文字の **c** であるため、両方を指定する必要はありません)。
- **T** は、この証明書がクライアント証明書に署名するための信頼できる認証局であることを示しています（は小文字の **c** であるため、**T** と **c**の両方を指定する必要はありません）。

各位置の信頼フラグを指定するには、文字をコンマで区切ります。たとえば、オプション **-t CT,C,c** は、クライアントおよびサーバーの TLS 証明書の署名、サーバーの電子メール証明書(S/MIME)の署名に信頼され、オブジェクトの署名に有効な CA であることを意味します（信頼されていません）。

- これにより、この証明書が別の証明書に署名し、それがオブジェクト署名に使用されると、その証明書が無効になります。

信頼なし（または信頼の欠如）は、**-t、、** を使用して指定できます。

データベース内のすべての証明書のトラストレベルを表示するには、以下を実行します。

- **certutil -L -d**
- 各証明書のニックネームが一覧表示され、行の最後に信頼フラグが指定されます。

-h オプションの HSM に関する注記を参照してください。

信頼レベルは、certutil の man ページで指定されていることに注意してください。このドキュメントを参照するには、certutil が正しくインストールされているシステムで **man certutil** コマンドを実行します。

-h <HSM>
-h <HSM> オプションは、 操作を実行する HSM の名前を指定します。

HSM は信頼を保存できないため、**-h** オプションは **-t** オプションと互換性がありません。信頼を保存できるのは NSS DB であるため、- **h <HSM> と一緒に certutil -A コマンドまたは certutil -M コマンドを使用すると** 失敗します。代わりに、**-h** オプションを指定せずに、別の **certutil -M** コマンドで必要な信頼レベルを指定します。

設定に応じて、この値を更新してください。

-e
-e オプションは、**certutil -V** コマンドとともに使用する場合に、署名の有効性もチェックされるように指定します。**PKICertImport** は、証明書署名の検証を常に実行し、**-e** オプションを理解しません。

-a
-a オプションは、問題のキーが PEM (ASCII)形式であることを指定します。

-i <certificate>
-i <certificate> オプションは、証明 書へのパスを指定します。これは、インポートする証明書へのパスを指定するために **certutil -A** コマンドでのみ使用されます。

設定に応じて、この値を更新してください。

-u <usage>
-u <usage&gt; オプションは、**certutil -V** コマンドと併用する際に検証する証明書の使用を指定します。

次のセクションで参照されるいくつかの使用法の文字があります。

- **-u C** は、クライアント TLS 証明書の検証を表します。これは証明書を許可しますが、有効期限と署名を確認することに注意してください。
- **-u V** は、サーバーの TLS 証明書の検証を表します。これにより CA 証明書が拒否され、有効期限と署名を確認することに注意してください。
- **-u L** は CA TLS 証明書の検証を表します。これにより信頼フラグが検証され(**c** が存在するか確認)、鍵の使用方法を確認して、キーが CA キーであることを確認します。これは、有効期限および署名も確認します。
- **-u O** は OCSP ステータスレスポンダー証明書の検証を表します。これにより、期限切れと署名を確認することに注意してください。
- **-u J** はオブジェクト署名証明書の検証を表します。これにより、期限切れと署名を確認することに注意してください。

誤った使用方法オプションが指定されているか、証明書の信頼フラグが正しくない場合(CA TLS 証明書の **c** フラグがないなど)、**certutil -V** は誤った結果を提供します。

注記

使用方法の詳細情報は、certutil の man ページに記載されています。このドキュメントを参照するには、certutil が正しくインストールされているシステムで **man certutil** コマンドを実行します。

11.2. ルート証明書のインポート

まず、ディレクトリーを NSS DB に変更します。

- **cd /path/to/nssdb**

これらの手順の実行中に Web サービスがオフラインになり (停止、無効化など) し、他のプロセス (ブラウザーなど) による NSS DB への同時アクセスがないことを確認します。これにより、NSS DB が破損したり、これらの証明書の使用が不適切になる場合があります。

新しいルート証明書をインポートする必要がある場合は、証明書がいくつもの証明書に署名できるため、安全な方法でこの証明書を取得するようにしてください。**ca_root.crt** という名前のファイルにすでに存在していることを前提とします。お好みのシナリオに合わせて、正しい名前とパスを置き換えます。

以下の **certutil** および **PKICertImport** オプションの詳細は、「**certutil および PKICertImport**」 を参照してください。

ルート証明書をインポートするには、次のコマンドを実行します。

- **PKICertImport -d . -n "CA Root" -t "CT,C,C" -a -i ca_root.crt -u L** コマンドを実行します。

  このコマンドは、ルート証明書を NSS DB に検証し、インポートします。エラーメッセージが

出力されず、戻りコードが 0 の場合は、検証が成功します。戻りコードを確認するには、上記のコマンド実行直後に **echo $?** を実行します。ほとんどの場合は、視覚的なエラーメッセージが出力されます。証明書は通常、有効期限が切れるか、または CA 証明書であるかから検証に失敗します。したがって、証明書ファイルが正しいことを確認し、最新の状態にしてください。発行者に連絡し、すべての中間証明書およびルート証明書がシステムに存在することを確認します。

11.3. 中間証明書チェーンのインポート

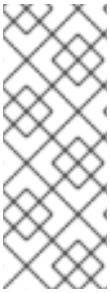
開始する前に、ディレクトリーを NSS DB に変更します。

- **cd /path/to/nssdb**

これらの手順の実行中に Web サービスがオフラインであることを確認します (停止、無効化など)、他のプロセス (ブラウザーなど) による NSS DB への同時アクセスがないことを確認します。これにより、NSS DB が破損したり、これらの証明書の使用が不適切になる場合があります。

ルート証明書をインポートして信頼していない場合は、「ルート証明書のインポート」を参照してください。

ルートサーバーとエンドサーバーまたはクライアント証明書間に一連の中間証明書が指定される場合は、ルート CA 証明書から最も適したように、署名済み証明書チェーンをインポートおよび検証する必要があります。中間 CA は **ca_sub_<num>.crt** という名前のファイルにあると仮定します（例：**ca_sub_1.crt**、**ca_sub_2.crt** など）。デプロイメントに合わせて、証明書の名前とパスを置き換えます。



注記

代わりに、**fullchain.crt**、**fullchain.pem**、または同様の名前の単一ファイルが指定されており、これには複数の証明書が含まれ、各ブロック(----BEGIN CERTIFICATE----- および -----END CERTIFICATE----- マーカーと -----END CERTIFICATE----- マーカーを含む)を独自のファイルにコピーして上記の形式に分割します。最初のものは **ca_sub_<num>.crt** という名前で、最後のものは **service.crt** という名前のサーバー証明書になります。サーバー証明書については、以降のセクションで説明します。

まず、ルート CA 証明書から最も遠い順に、中間 CA をインポートして検証します。存在していない場合は、次のセクションに進みます。

以下の **certutil** および **PKICertImport** オプションの詳細は、「**certutil** および **PKICertImport**」 を参照してください。

チェーン内のすべての中間証明書に対して、以下を行います。

- Execute **PKICertImport -d . -n "CA Sub $num" -t "CT,C,C" -a -i ca_sub_$num.crt -u L**

  このコマンドは、中間 CA 証明書を NSS DB に検証し、インポートします。エラーメッセージが出力されず、戻りコードが 0 の場合は、検証が成功します。戻りコードを確認するには、上記のコマンド実行直後に **echo $?** を実行します。ほとんどの場合は、視覚的なエラーメッセージが出力されます。検証に成功しなかった場合は、発行者に連絡し、すべての中間証明書およびルート証明書がシステムに存在することを確認します。

11.4. NSS データベースでの証明書のインポート

これらの手順の実行中に Web サービスがオフラインになり (停止、無効化など) し、他のプロセス (ブラウザーなど) による NSS データベースへの同時アクセスがないことを確認します。これにより、NSS データベースが破損したり、これらの証明書の使用が不適切になる場合があります。

従う手順セットは、問題の証明書の使用により異なります。

- サブシステムの **auditSigningCert** については、以下の手順に従って **オブジェクト署名証明書** を検証してください。
- CA サブシステムの **caSigningCert** については、**中間証明書チェーン** をインポートして検証するための上記の手順に従いますが、caSigningCert でのみ行います。
- CA サブシステムの **ocspSigningCert** については、以下の手順に従って **OCSP 証明書** を検証してください。
- ユーザーのクライアントまたは S/MIME 証明書の場合は、**Client Certificate の手順** を実行します。

以下の **certutil** および **PKICertImport** オプションの詳細は、「**certutil および PKICertImport**」 を参照してください。

NSS データベースへのクライアント証明書のインポート

NSS データベースへのクライアント証明書のインポートするには、以下を実行します。

1. NSS データベースディレクトリーに移動します。以下に例を示します。

```
# cd /path/to/nssdb/
```

2. ルート証明書をまだインポートおよび信頼していない場合は、インポートして信頼します。詳細は、「ルート証明書のインポート」を参照してください。
3. 中間証明書をインポートおよび検証していない場合は、中間証明書をインポートおよび検証します。詳細は、「中間証明書チェーンのインポート」を参照してください。
4. クライアント証明書を検証し、インポートします。

```
# PKICertImport -d . -n "client name" -t ",," -a -i client.crt -u C
```

エラーメッセージが出力されず、戻りコードが 0 の場合は、検証が成功します。戻りコードを確認するには、上記のコマンド実行直後に **echo $?** を実行します。ほとんどの場合は、視覚的なエラーメッセージが出力されます。検証に成功しなかった場合は、発行者に連絡し、すべての中間証明書およびルート証明書がシステムに存在することを確認します。

オブジェクト署名証明書のインポート

オブジェクト署名証明書をインポートするには、以下を実行します。

1. NSS データベースディレクトリーに移動します。以下に例を示します。

```
# cd /path/to/nssdb/
```

2. ルート証明書をまだインポートおよび信頼していない場合は、インポートして信頼します。詳細は、「ルート証明書のインポート」を参照してください。
3. 中間証明書をインポートおよび検証していない場合は、中間証明書をインポートおよび検証します。詳細は、「中間証明書チェーンのインポート」を参照してください。

4. オブジェクト署名証明書を検証し、インポートします。

```
# PKICertImport -d . -n "certificate name" -t ",,P" -a -i objectsigning.crt -u J
```

エラーメッセージが出力されず、戻りコードが 0 の場合は、検証が成功します。戻りコードを確認するには、上記のコマンド実行直後に **echo $?** を実行します。ほとんどの場合は、視覚的なエラーメッセージが出力されます。検証に成功しなかった場合は、発行者に連絡し、すべての中間証明書およびルート証明書がシステムに存在することを確認します。

OCSP レスポンダーのインポート

OCSP レスポンダーをインポートするには、以下を行います。

1. NSS データベースディレクトリーに移動します。以下に例を示します。

```
# cd /path/to/nssdb/
```

2. ルート証明書をまだインポートおよび信頼していない場合は、インポートして信頼します。詳細は、「ルート証明書のインポート」を参照してください。

3. 中間証明書をインポートおよび検証していない場合は、中間証明書をインポートおよび検証します。詳細は、「中間証明書チェーンのインポート」を参照してください。

4. OCSP レスポンダー証明書を検証およびインポートします。

```
# PKICertImport -d . -n "certificate name" -t ",," -a -i ocsp.crt -u O
```

エラーメッセージが出力されず、戻りコードが 0 の場合は、検証が成功します。戻りコードを確認するには、上記のコマンド実行直後に **echo $?** を実行します。ほとんどの場合は、視覚的なエラーメッセージが出力されます。検証に成功しなかった場合は、発行者に連絡し、すべての中間証明書およびルート証明書がシステムに存在することを確認します。

第12章 証明書システムユーザーおよびグループの管理

本章では、管理、エージェントサービス、およびエンドエンティティーページにアクセスするための承認を設定する方法を説明します。

12.1. 認可について

注記

このセクションで説明する各グループに関連付けられている ACL は変更しないでください。

承認 は、Certificate System に関連付けられた特定のタスクにアクセスできるようにするプロセスです。アクセスは、特定のユーザーまたはグループのサブシステムの特定領域に対応し、異なるユーザーおよびグループに対して異なるタスクを許可できるように制限できます。

ユーザーは、作成されるサブシステムに固有のものです。各サブシステムには、インストールされている他のサブシステムから独立した独自のユーザーセットがあります。ユーザーはグループに配置され、事前定義またはユーザーの作成が可能です。**アクセス制御リスト** (ACL) で、権限がグループに割り当てられます。管理コンソール、エージェントサービスインターフェイス、およびエンドエンティティーページの領域に関連付けられた ACL があり、操作の続行を許可する前に許可チェックを実行します。各 ACL の**アクセス制御命令** (ACI) が作成され、その ACL が指定されたユーザー、グループ、または IP アドレスに対して許可される操作を許可または拒否します。

ACL には、作成されるデフォルトグループのデフォルト ACI セットが含まれます。

承認は以下のプロセスを経て行われます。

1. ユーザーは、証明書を使用してインターフェイスに対して認証を行います。
2. サーバーは、証明書をデータベースに保存されている証明書と照合して、ユーザーを認証します。サーバーはまた、証明書が有効であることを確認し、証明書の DN をユーザーに関連付けてユーザーエントリーを確認することにより、ユーザーのグループメンバーシップを見つけます。
3. ユーザーが操作を実行しようとすると、承認メカニズムはユーザーのユーザー ID、ユーザーが属するグループ、ユーザーの IP アドレスを、そのユーザー、グループ、または IP アドレスに設定された ACL と比較します。その操作を許可する ACL が存在する場合、操作は続行されます。

12.2. デフォルトグループ

ユーザーの権限は、ユーザーのグループ (ロール) メンバーシップにより決定されます。ユーザーを割り当てることのできる 3 つのグループ (ロール) があります。

- **管理者**。このグループには、管理インターフェイスで利用可能なすべてのタスクへの完全なアクセスが付与されます。
- **エージェント**。このグループには、エージェントサービスインターフェイスで利用可能なすべてのタスクに完全アクセスできます。
- **監査者**。このグループには、署名済み監査ログを表示するためのアクセスが付与されます。このグループには他の権限がありません。

サブシステム間の通信のみに限り作成される 4 つ目のロールがあります。管理者は、実際のユーザーをこのようなロールに割り当てることはできません。

- **エンタープライズ管理者**。各サブシステムインスタンスには、設定中にセキュリティードメインに参加していると、エンタープライズ管理者としてサブシステム固有のロールが自動的に割り当てられます。これらのロールはセキュリティードメインのサブシステム間で信頼できる関係を自動的に提供し、各サブシステムが他のサブシステムと効率的に対話できるようにします。

12.2.1. 管理者

管理者には、すべての管理タスクを実行できるパーミッションがあります。ユーザーは、グループの **Administrators** グループに追加され、管理者として特定されます。このグループの各メンバーには、Certificate System のそのインスタンスに対する管理権限が必要です。

Certificate System インスタンスごとに少なくとも 1 つの管理者を定義する必要がありますが、インスタンスに割り当てることのできる管理者の数に制限はありません。インスタンスの設定時に最初の管理者エントリーが作成されます。

管理者は、Certificate System ユーザー ID とパスワードを使用して単純なバインドで認証されます。

表12.1 セキュリティードメインのユーザーロール

ロール	説明
セキュリティードメインの管理者	• セキュリティードメインのユーザーおよびグループデータベースでユーザーを追加および変更します。 • 共有信頼ポリシーを管理します。 • ドメインサービスのアクセス制御を管理します。 デフォルトでは、ドメインをホストする CA の CA 管理者はセキュリティードメイン管理者として割り当てられます。
エンタープライズ CA 管理者	• ドメインのサブ CA、サーバー、およびサブシステムの証明書を自動的に承認します。 • セキュリティードメインで CA サブシステム情報を登録および登録解除します。
エンタープライズ KRA 管理者	• ドメインの CA からトランスポート、ストレージ、サーバー、およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで KRA サブシステム情報を登録および登録解除します。 • KRA コネクター情報を CA にプッシュします。

ロール	説明
エンタープライズ OCSP 管理者	• ドメイン内の CA から OCSP、サーバー、およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで OCSP サブシステム情報を登録および登録解除します。 • CRL を公開する情報を CA にプッシュします。
エンタープライズ TKS 管理者	• ドメインの CA からサーバー証明書およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで TKS サブシステム情報を登録および登録解除します。
エンタープライズ TPS 管理者	• ドメインの CA からサーバー証明書およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで TPS サブシステム情報を登録および登録解除します。

必要に応じて、セキュリティードメイン管理者はセキュリティードメインおよび個別のサブシステムでアクセス制御を管理できます。たとえば、セキュリティードメイン管理者はアクセスを制限することで、KRA の管理者のみが 座有無部門の KRA を設定できるようにすることができます。

Enterprise サブシステムの管理者は、ドメインのサブシステムで操作を実行するのに十分な特権が付与されます。たとえば、エンタープライズ CA の管理者は、設定中にサブ CA 証明書を自動的に承認する特権があります。セキュリティードメイン管理者は、必要に応じてこの適切な制限を行うことができます。

12.2.2. 監査者

監査人は、署名された監査ログを表示でき、システムの動作を監査するために作成されます。監査人はサーバーを管理することはできません。

監査人は、ユーザーを **Auditors** グループに追加して、監査人の証明書をユーザーエントリーに保存することによって作成されます。監査人の証明書は、監査ログの署名に使用されるキーペアの秘密キーを暗号化するために使用されます。

サブシステムの設定時に **Auditors** グループが設定されます。設定中、このグループには監査人は割り当てられません。

12.2.3. エージェント

エージェントは、エンドエンティティー証明書と鍵管理の特権が割り当てられているユーザーです。エージェントは、エージェントサービスインターフェイスにアクセスできます。

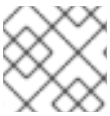
エージェントは、ユーザーを適切なサブシステムエージェントグループに割り当て、エージェントからの要求を処理するためにサブシステムへの TLS クライアント認証にエージェントが使用する必要のある証明書を識別することによって作成されます。各サブシステムには独自のエージェントグループがあります。

- 証明書マネージャーエージェントグループ。
- キーリカバリー認証局エージェントグループ。
- オンライン証明書ステータスマネージャーエージェントグループ
- トークンキーサービスエージェントのグループ。
- Token Processing System Agents グループ。

各 Certificate System サブシステムには、サブシステムで定義されたロールを持つ独自のエージェントがあります。各サブシステムには少なくとも 1 つのエージェントが必要ですが、サブシステムを持つエージェントの数に制限はありません。

Certificate System は、内部データベース内でユーザーの TLS クライアント証明書をチェックして、エージェント権限を持つユーザーを特定し、認証します。

12.2.4. エンタープライズグループ



注記

このグループに実際のユーザーを割り当てることはできません。

サブシステムの設定中に、すべてのサブシステムインスタンスがセキュリティードメインに参加します。各サブシステムインスタンスには、サブシステム固有のロールがエンタープライズ管理者として自動的に割り当てられます。これらのロールはセキュリティードメインのサブシステム間で信頼できる関係を自動的に提供し、各サブシステムが他のサブシステムと効率的に対話できるようにします。たとえば、これにより、OCSP はドメイン内のすべての CA に CRL 公開情報をプッシュし、KRA は KRA コネクター情報をプッシュし、CA は CA 内で生成された証明書を自動的に承認します。

Enterprise サブシステムの管理者は、ドメインのサブシステムで操作を実行するのに十分な特権が付与されます。各サブシステムには独自のセキュリティードメインロールがあります。

- エンタープライズ CA 管理者
- エンタープライズ KRA 管理者
- エンタープライズ OCSP 管理者
- エンタープライズ TKS 管理者
- エンタープライズ TPS 管理者

また、ドメイン内のセキュリティードメイン、アクセス制御、ユーザー、および信頼関係を管理する CA インスタンス用の Security Domain Administrators のグループもあります。

各サブシステム管理者は、セキュリティードメイン CA によって設定時に発行されたサブシステム証明書を使用した TLS クライアント認証を使用して他のサブシステムに対して認証します。

12.3. CA、OCSP、KRA、または TKS のユーザーおよびグループの管理

ユーザーが実行できる操作の多くは、ユーザーが属するグループによって決定されます。たとえば、CA のエージェントは証明書とプロファイルを管理し、管理者は CA サーバーの設定を管理します。

4 つのサブシステム (CA、OCSP、KRA、および TKS) Java 管理コンソールを使用してグループとユーザーを管理します。TPS には Web ベースの管理サービスがあり、ユーザーおよびグループは Web サービスページで設定されます。

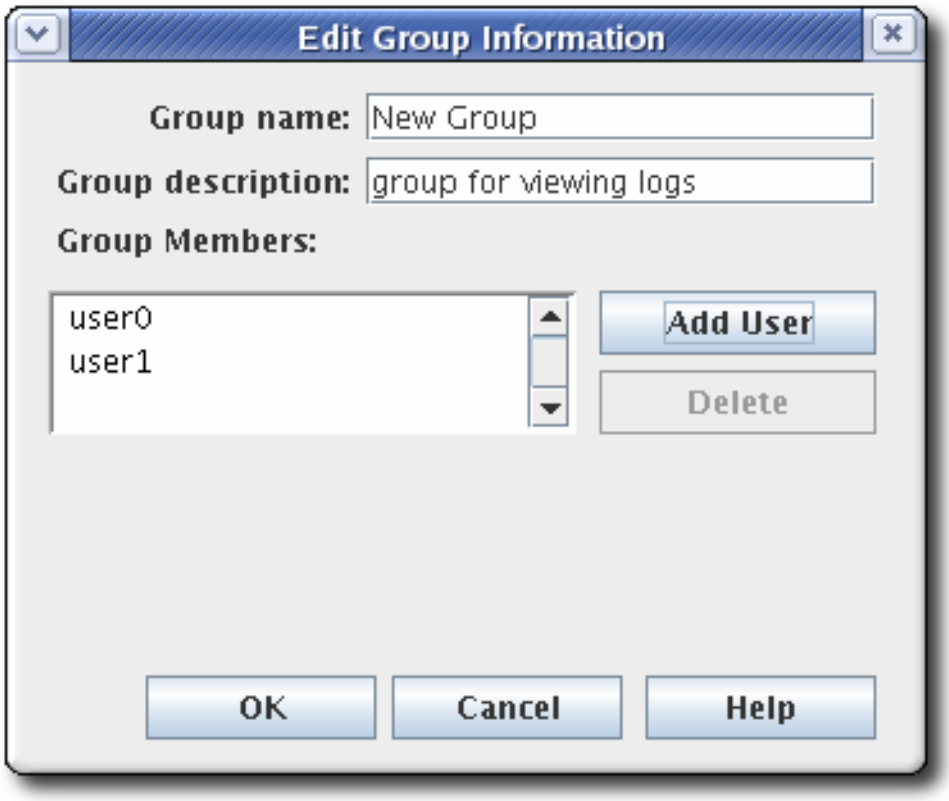
12.3.1. グループの管理

12.3.1.1. 新規グループの作成

1. 管理コンソールにログインします。

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

2. 左側のナビゲーションメニューから **Users and Groups** を選択します。
3. **Groups** タブを選択します。
4. **Edit** をクリックして、グループ情報を入力します。



   内部データベースにすでに存在するユーザーのみを追加することが可能です。

5. ACL を編集して、グループの権限を付与します。詳細は、「ACL の編集」を参照してください。グループの ACL に ACI が追加されていない場合、グループには Certificate System の一部にアクセスパーミッションがありません。

12.3.1.2. グループのメンバーの変更

すべてのグループからメンバーを追加または削除できます。管理者のグループには、最低でもユーザーエントリーが 1 つ必要です。

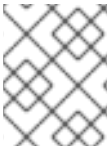
1. 管理コンソールにログインします。
2. 左側のナビゲーションツリーから **Users and Groups** を選択します。
3. **Groups** タブをクリックします。
4. 名前の一覧からグループを選択し、**Edit** をクリックします。
5. 適切な変更を加えます。
   - グループの説明を変更するには、**Group description** フィールドに新しい説明を入力します。
   - グループからユーザーを削除するには、ユーザーを選択し、**Delete** をクリックします。
   - ユーザーを追加するには、**Add User** をクリックします。ダイアログボックスから追加するユーザーを選択し、**OK** をクリックします。

12.3.2. ユーザーの管理 (管理者、エージェント、および監査者)

各サブシステムのユーザーは別々に維持されます。あるサブシステムの管理者であるからといって、その人が別のサブシステムに対する権限 (またはユーザーエントリー) を持っているとは限りません。ユーザーを設定して、ユーザー証明書を使用してサブシステムのエージェント、管理者、または監査担当者として信頼できます。

12.3.2.1. ユーザーの作成

Certificate System をインストールしたら、セットアップ時に作成したユーザーのみが存在します。本セクションでは、追加のユーザーを作成する方法を説明します。



注記

セキュリティー上の理由と監査証跡のため、Certificate System のユーザーと管理者には個別のアカウントを作成してください。

12.3.2.1.1. コマンドラインでのユーザーの作成

コマンドラインでユーザーを作成するには、以下を実行します。

1. ユーザーアカウントを追加します。たとえば、**example** ユーザーを CA に追加するには、以下を実行します。

```
# pki -c password -n caadmin \
    ca-user-add example --fullName "Example User"
---------------------
Added user "example"
---------------------
  User ID: example
  Full name: Example User
```

このコマンドは、**caadmin** ユーザーを使用して新規アカウントを追加します。

2. 必要に応じて、グループにユーザーを追加します。たとえば、**example** ユーザーを **Certificate Manager Agents** グループに追加するには、次のコマンドを実行します。

```
# pki -p password -n "caadmin" \
    user-add-membership example Certificate Manager Agents
```

3. 証明書要求を作成します。

- Certificate System 環境に Key Recovery Authority (KRA) が存在する場合は、以下を行います。

```
# 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
```

このコマンドは、証明書署名要求 (CSR) を ***~/user_name.req*** ファイルに **CRMF** 形式を保存します。

- 証明書システム環境に Key Recovery Authority (KRA) が存在しない場合は、以下を行います。

```
# PKCS10Client -d ~/.dogtag/pki-instance_name/ -p password \
    -n "user_name" -o ~/user_name.req
```

このコマンドは、CSR を **pkcs10** 形式で **、 *~/user_name.req*** ファイルに保存します。

4. 登録リクエストを作成します。

a. ***~/cmc.role_crmf.cfg*** ファイルを以下の内容で作成します。

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

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
#Multiple files are supported. They must be separated by space.
input=~/user_name.req

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

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

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
nickname=PKI Administrator for Example.com

#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=~/.dogtag/pki-instance_name/

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

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

直前の手順で使用した環境および CSR 形式に基づいて、パラメーターを設定します。

b. 以前に作成した設定ファイルを **CMCRequest** ユーティリティーに渡して、CMC 要求を作成します。

```
# CMCRequest ~/cmc.role_crmf.cfg
```

5. CMS (CMC) 要求で Certificate Management を送信します。

a. ***~/HttpClient_role_crmf.cfg*** ファイルを以下の内容で作成します。

```
# #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 TLS client authentication cert can be found (default is
internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=~/.dogtag/pki-instance_name/

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

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

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=PKI Administrator for Example.com

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitCMCFull
```

環境に応じてパラメーターを設定します。

b. CA に要求を送信します。

```
# 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
MIIJ1wYJKoZIhvcNAQcCoIIJyDCCCcQCAQMxDzANBglghkgBZQMEAgEFADAxBggr
...
The response in data format is stored in ~/cmc.role_crmf.resp
```

c. 結果を確認します。

```
# 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. 必要に応じて、証明書をユーザー自身の ~/**.dogtag**/**pki-instance_name**/ データベースにインポートするには、次のコマンドを実行します。

```
# certutil -d ~/.dogtag/pki-instance_name/ -A -t "u,u,u" -n "user_name certificate" -i
~/cmc.role_crmf.resp
```

7. ユーザーレコードに証明書を追加します。

a. ユーザーのシリアル番号を検出できる証明書を一覧表示します。たとえば、証明書のサブジェクトに **example** ユーザー名が含まれる証明書を一覧表示するには、次のコマンドを実行します。

```
pki -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
```

次の手順では、証明書のシリアル番号が必要です。

b. シリアル番号を使用して、証明書リポジトリーから Certificate System データベースのユーザーアカウントに証明書を追加します。たとえば、CA ユーザーの場合は以下を実行します。

```
pki -c password -n caadmin ca-user-cert-add example --serial 0x6
```

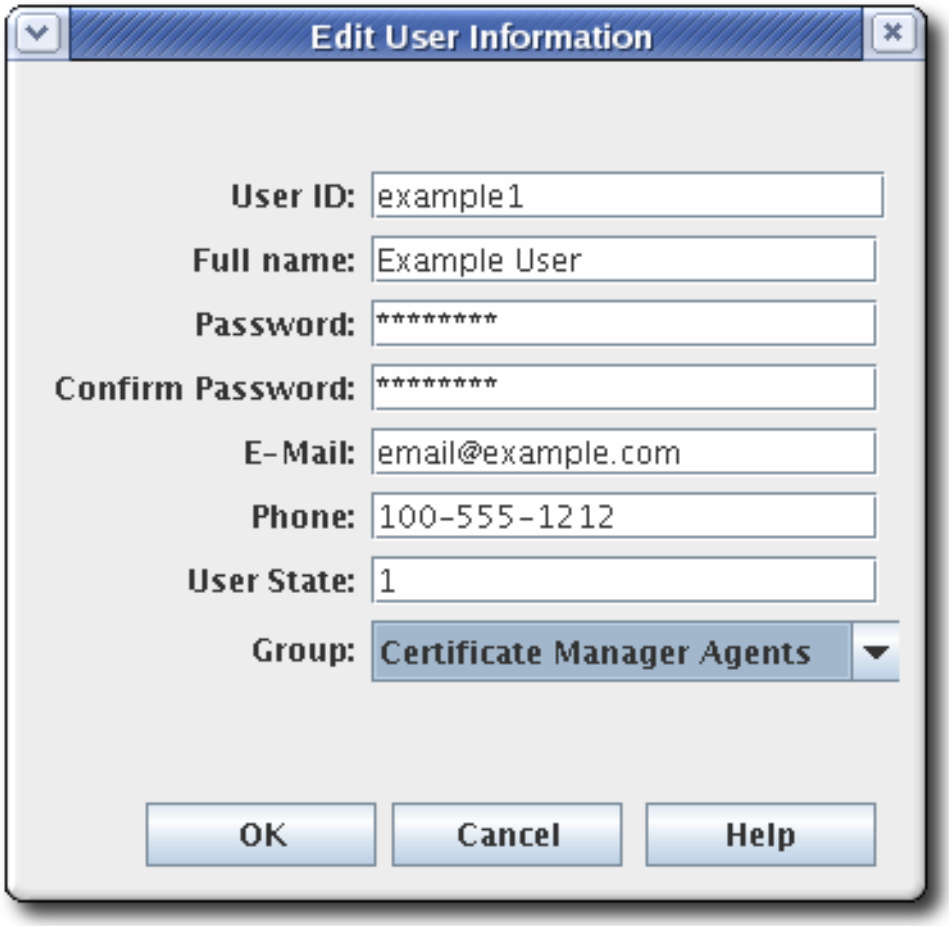
12.3.2.1.2. コンソールを使用したユーザーの作成

PKI コンソールを使用してユーザーを作成するには、次のコマンドを実行します。

1. 管理コンソールにログインします。

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

2. **Configuration** タブで、**Users and Groups** を選択します。**Add** をクリックします。
3. **Edit User Information** ダイアログに情報を入力します。



情報のほとんどは、ユーザー名、メールアドレス、パスワードなどの標準のユーザー情報で

す。このウィンドウには、**User State** と呼ばれるフィールドも含まれ、このフィールドには、ユーザーに関する追加情報を追加するのに使用される文字列を含めることができます。ほとんどの場合、このフィールドは、アクティブユーザーであるかどうかを確認できます。

4. ユーザーが属するグループを選択します。ユーザーのグループメンバーシップは、ユーザーが持つ特権を決定します。エージェント、管理者、および監査人を適切なサブシステムグループに割り当てます。

5. ユーザーの証明書を保存します。

    a. CA エンドエンティティーサービスページでユーザー証明書を要求します。

    b. ユーザープロファイルに対して自動登録が設定されていない場合は、証明書要求を承認します。

    c. 通知メールで提供される URL を使用して証明書を取得し、base-64 でエンコードされた証明書をローカルファイルまたはクリップボードにコピーします。

    d. 新しいユーザーエントリーを選択し、**Certificates** をクリックします。

    e. **Import** をクリックし、Base-64 でエンコードされた証明書に貼り付けます。

12.3.2.2. 証明書システムユーザー証明書の変更

1. 管理コンソールにログインします。

2. **Users and Groups** を選択します。

3. ユーザー ID の一覧から編集するユーザーを選択し、**Certificates** をクリックします。

4. **Import** をクリックして、新しい証明書を追加します。

5. **Import Certificate** ウィンドウで、テキストエリアに新しい証明書を貼り付けます。**-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** マーカー行を含めます。

12.3.2.3. 管理者、エージェント、および監査ユーザー証明書の更新

証明書を更新する方法は 2 つあります。証明書を **再生成** すると、元の鍵と元のプロファイルと要求を取得し、新しい有効期間と有効期限で同一の鍵を再作成します。証明書の**キーを再入力** すると、最初の証明書要求が元のプロファイルに再送信されますが、新しいキーペアが生成されます。管理者証明書は、キーを再入力することで更新できます。

各サブシステムには、サブシステムの作成時に作成されたブートストラップ管理者ユーザーがいます。デフォルトの更新プロファイルの 1 つを使用して、元の証明書の有効期限が切れる前に、このユーザーに新しい証明書を要求できます。

管理ユーザーの証明書は、元の証明書のシリアル番号を使用して、エンドユーザー登録フォームで直接更新できます。

1. 管理ユーザー証明書を更新します。詳細は「証明書の更新」 を参照してください。

2. 更新されたユーザー証明書を内部 LDAP データベースのユーザーエントリーに追加します。

    a. サブシステムのコンソールを開きます。

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

b. 設定 | ユーザーとグループ | ユーザー | 管理 | 証明書 | インポート

c. **Configuration** タブで、**Users and Groups** を選択します。

d. **Users** タブで、更新された証明書でユーザーエントリーをダブルクリックして、**Certificates** をクリックします。

e. **Import** をクリックし、Base-64 でエンコードされた証明書に貼り付けます。

これは、**ldapmodify** を使用して、**uid=admin,ou=people,dc=subsystem-base-DN** など、ユーザーエントリーの ***userCertificate*** 属性を置き換え、内部の LDAP データベースでユーザーエントリーに直接更新した証明書を追加しました。

12.3.2.4. 証明書システムユーザーの削除

ユーザーは内部データベースから削除できます。内部データベースからユーザーを削除すると、そのユーザーが属するすべてのグループから削除されます。特定のグループからユーザーを削除するには、グループメンバーシップを変更します。

以下の手順を実行して、内部データベースから特権ユーザーを削除します。

1. 管理コンソールにログインします。
2. 左側のナビゲーションメニューから **Users and Groups** を選択します。
3. ユーザー ID の一覧からユーザーを選択して、**Delete** をクリックします。
4. プロンプトが表示されたら、削除を確認します。

12.4. ユーザーのアクセス制御の設定

承認 は、ユーザーが操作を実行できるかどうかを確認するメカニズムです。許可ポイントは、許可チェックを必要とする特定の操作グループで定義されます。

12.4.1. アクセス制御について

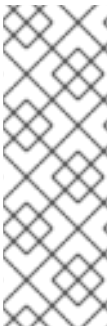
アクセス制御リスト (ACL) は、サーバー操作への承認を指定するメカニズムです。承認チェックが実行される操作ごとに ACL が存在します。ACL に追加の操作を追加できます。

ACL には、読み取りや変更などの操作を具体的に許可または拒否する **アクセス制御命令** (ACI) が含まれます。ACI にはエバリュエーターの式も含まれます。ACL のデフォルトの実装は、ユーザー、グループ、および IP アドレスのみを、可能なエバリュエータータイプとして指定します。ACL の各 ACI は、アクセスが許可または拒否されるかどうか、特定の Operator が許可または拒否されているか、および操作を実行するためのユーザー、グループ、または IP アドレスが許可または拒否されるかどうかを指定します。

Certificate System ユーザーの特権は、ユーザーがメンバーであるグループ、ユーザー自身、またはユーザーの IP アドレスに関連付けられているアクセス制御リスト (ACL) を変更することによって変更されます。新規グループは、そのグループをアクセス制御リストに追加することで、アクセス制御リストに割り当てられます。たとえば、ログ **LogAdmins** の表示が許可される管理者用の新規グループは、このグループの読み取りまたは修正を許可するためにログに関連する ACL に追加できます。このグループが他の ACL に追加されない場合、このグループのメンバーはログにのみアクセスできます。

ACL の ACI エントリーを編集して、ユーザー、グループ、または IP アドレスへのアクセスが変更されます。ACL インターフェイスでは、各 ACI が独自の行に表示されます。このインターフェイスウィンドウで、ACI の構文は以下のとおりです。

```
allow|deny (operation) user|group|IP="name"
```



注記

IP アドレスは、IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:** および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。

たとえば、以下は ACI で、管理者は読み取り操作を実行できます。

```
allow (read) group="Administrators"
```

ACI には、複数の操作またはアクションを設定できます。操作は、両側にスペースを入れずにコンマで区切ります。以下に例を示します。

```
allow (read,modify) group="Administrators"
```

ACI は、2 つのパイプ記号で区切ることにより、複数のグループ、ユーザー、または IP アドレスを、両側にスペースがある状態で指定することができます (||)。以下に例を示します。

```
allow (read) group="Administrators" || group="Auditors"
```

管理コンソールは ACI を作成または変更できます。このインターフェイスは、**Allow and Deny** フィールドで操作を許可するかどうか、**Operations** フィールドで可能な操作を設定し、次に **Syntax** フィールドでグループ、ユーザー、または IP アドレスを一覧表示します。

ACI は指定されたグループ、ユーザー ID、または IP アドレスの操作を許可または拒否できます。通常、アクセスを拒否するために ACI を作成する必要はありません。ユーザー ID、グループ、または IP アドレスを含む allow ACI がない場合、グループ、ユーザー ID、または IP アドレスへのアクセスは拒否されます。

注記

ユーザーがリソースのどの操作にも明示的に許可されていない場合、このユーザーは拒否されます。アクセスを拒否する必要はありません。

たとえば、ユーザー JohnB は **Administrators** グループのメンバーです。ACL には以下の ACL のみがある場合は、allow ACI に一致しないため、JohnB はすべてのアクセスを拒否します。

```
Allow (read,modify) group="Auditors" || user="BrianC"
```

通常、deny ステートメントを含める必要はありません。ただし、指定すると便利な場合もあります。たとえば、**Administrators** グループのメンバーである **JohnB** が唯一実行されています。ユーザーをすぐに削除できない場合は、特に **JohnB** へのアクセスを拒否する必要がある場合があります。もう 1 つの状況は、ユーザー **BrianC** が管理者であるが、一部のリソースを変更する権限を持たない場合です。**Administrators** グループはこのリソースにアクセスする必要があるため、**BrianC** はこのユーザーアクセスを拒否する ACI を作成して、アクセスを拒否することができます。

許可される権限は、ACI が操作の実行を許可または拒否することで ACI が制御する操作です。ACL に設定できるアクションは ACL とサブシステムによって異なります。定義できる 2 つの一般的な操作は、読み取りと変更です。

ACI エディターの構文フィールドは、式にエバリュエーターを設定します。エバリュエーターは、グループ、名前、および IP アドレス (IPv4 アドレスと IPv6 アドレスの両方) を指定できます。これらは、同一 (**=**) または非同一 (**!=**) として設定されたエンティティーの名前とともに指定されます。

ACL にグループを追加する構文は **group="groupname"** です。グループを除外する構文は **group!="groupname"** で、named グループ以外のグループを許可します。以下に例を示します。

```
group="Administrators" || group!="Auditors"
```

アスタリスク (*) などのワイルドカード文字を使用するなど、正規表現を使用してグループを指定することもできます。以下に例を示します。

```
group="* Managers"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

ACL にユーザーを追加する構文は **user="userID"** です。ユーザーを除外する構文は **user!="userID"** です。これにより、名前が指定されたユーザー ID 以外のユーザー ID も使用できます。以下に例を示します。

```
user="BobC" || user!="JaneK"
```

すべてのユーザーを指定するには、**anybody** の値を指定します。以下に例を示します。

```
user="anybody"
```

正規表現を使用して、アスタリスク (*) などのワイルドカード文字を使用するなど、ユーザー名を指定することもできます。以下に例を示します。

```
user="*johnson"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

ACL に IP アドレスを追加する構文は **ipaddress="ipaddress"** です。ACL から ID アドレスを除外する構文は **ipaddress!="ipaddress"** です。IP アドレスは数値を使用して指定します。DNS 値は許可されません。以下に例を示します。

```
ipaddress="12.33.45.99"
ipaddress!="23.99.09.88"
```

IP アドレスは、上記のように IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスには、ネットマスクが **n.n.n.n** または **n.n.n.n,m.m.m.m** の形式があります。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。以下に例を示します。

```
ipaddress="0:0:0:0:0:0:13.1.68.3"
```

正規表現を使用して、アスタリスク (*) などのワイルドカード文字を使用するなど、IP アドレスを指定することもできます。以下に例を示します。

```
ipaddress="12.33.45.*"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

各値を 2 つのパイプ文字 (||) で区切り、両側にスペースを入れることで、複数の値を持つ文字列を作成できます。以下に例を示します。

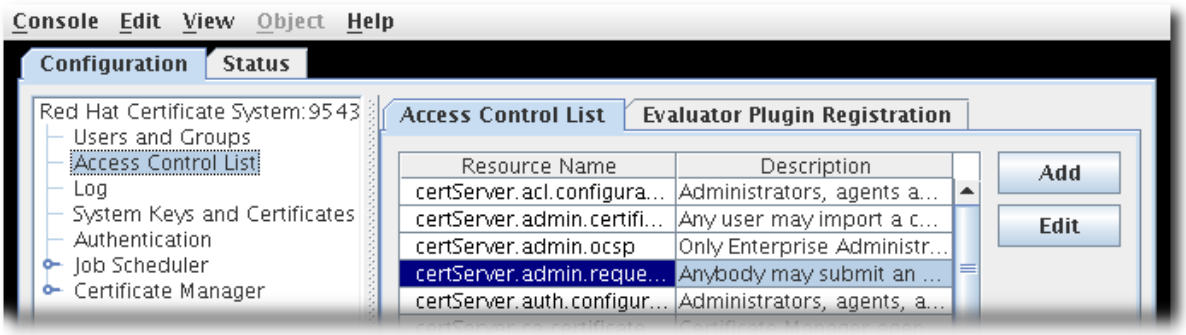
```
user="BobC" || group="Auditors" || group="Administrators"
```

12.4.2. ACL の追加

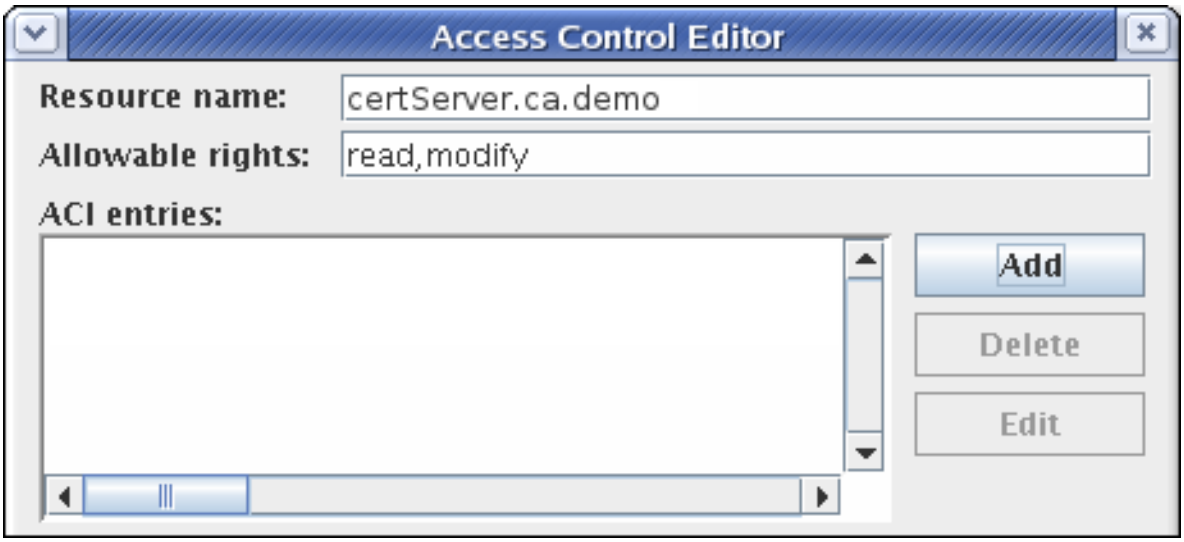
ACL は内部データベースに保存され、管理コンソールでのみ変更できます。

新しい ACL を追加するには、以下を実行します。

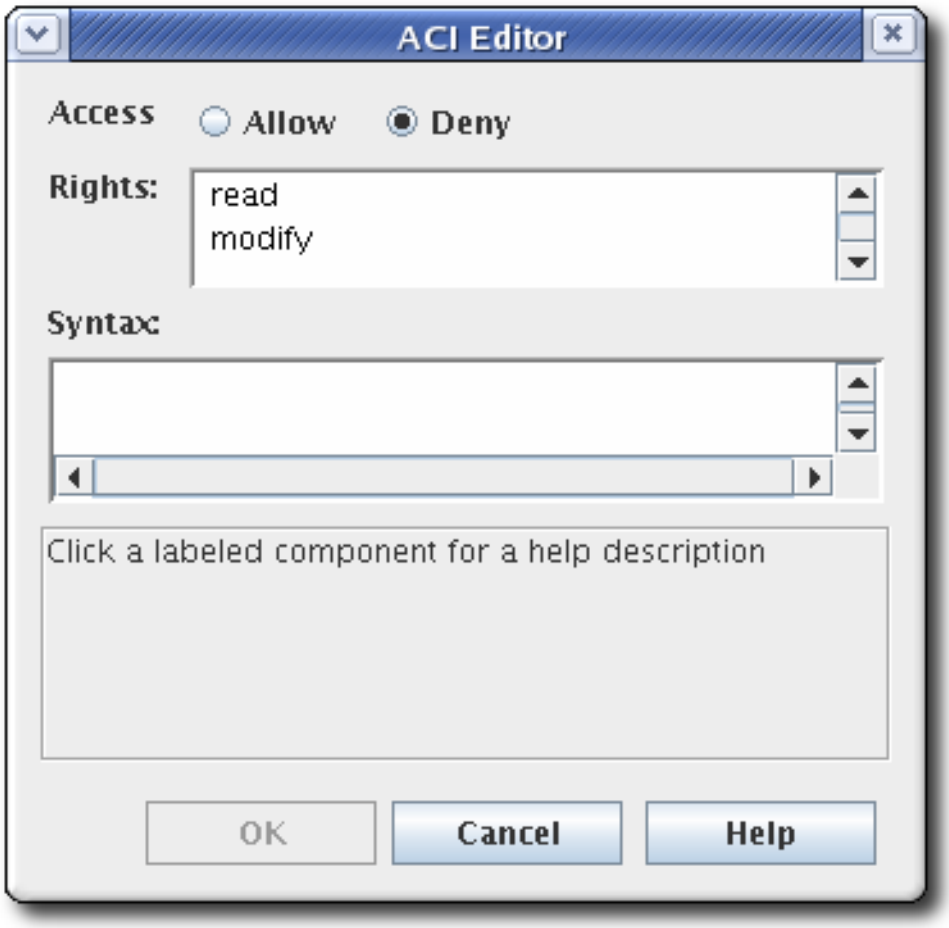
1. 管理コンソールにログインします。
2. **Access Control List** を選択します。



3. **Add** をクリックして、**Access Control Editor** を開きます。
4. **Resource name** および **Available rights** フィールドに入力します。



5. アクセス制御指示 (ACI) を追加するには、**Add** をクリックし、ACI 情報を提供します。



a. 指定したグループ、ユーザー、または IP アドレスへの操作を許可または拒否するには、**Access** フィールドから allow または deny ラジオボタンを選択します。アクセスの許可または拒否に関する詳細は、「アクセス制御について」を参照してください。

b. 権限を設定します。利用できるオプションは、**read** および **modify** です。両方を選択するには、エントリーの選択中に **Ctrl** ボタンまたは **Shift** ボタンを保持します。

c. **Syntax** フィールドでアクセスを許可または拒否されるユーザー、グループ、または IP アドレスを指定します。構文の詳細は、「アクセス制御について」を参照してください。

6. **OK** をクリックして、**Access Control Editor** 画面に戻ります。

7. **OK** をクリックして ACI を保存します。

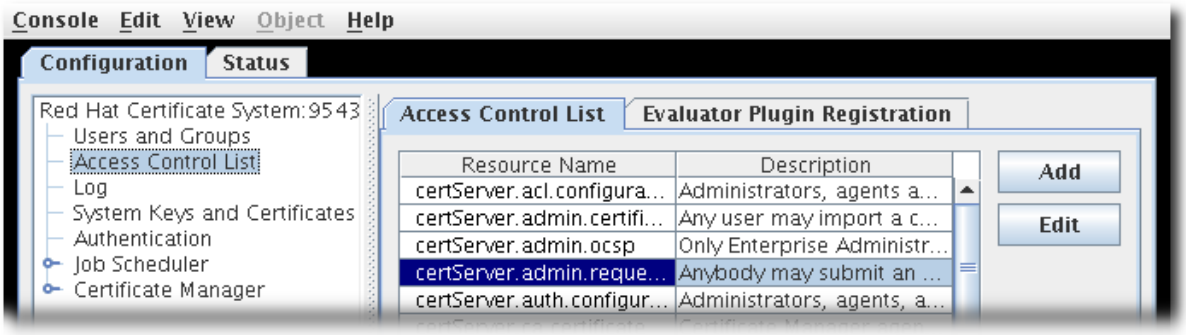
12.4.3. ACL の編集

ACL は内部データベースに保存され、管理コンソールでのみ変更できます。

既存の ACL を編集するには、以下を実行します。

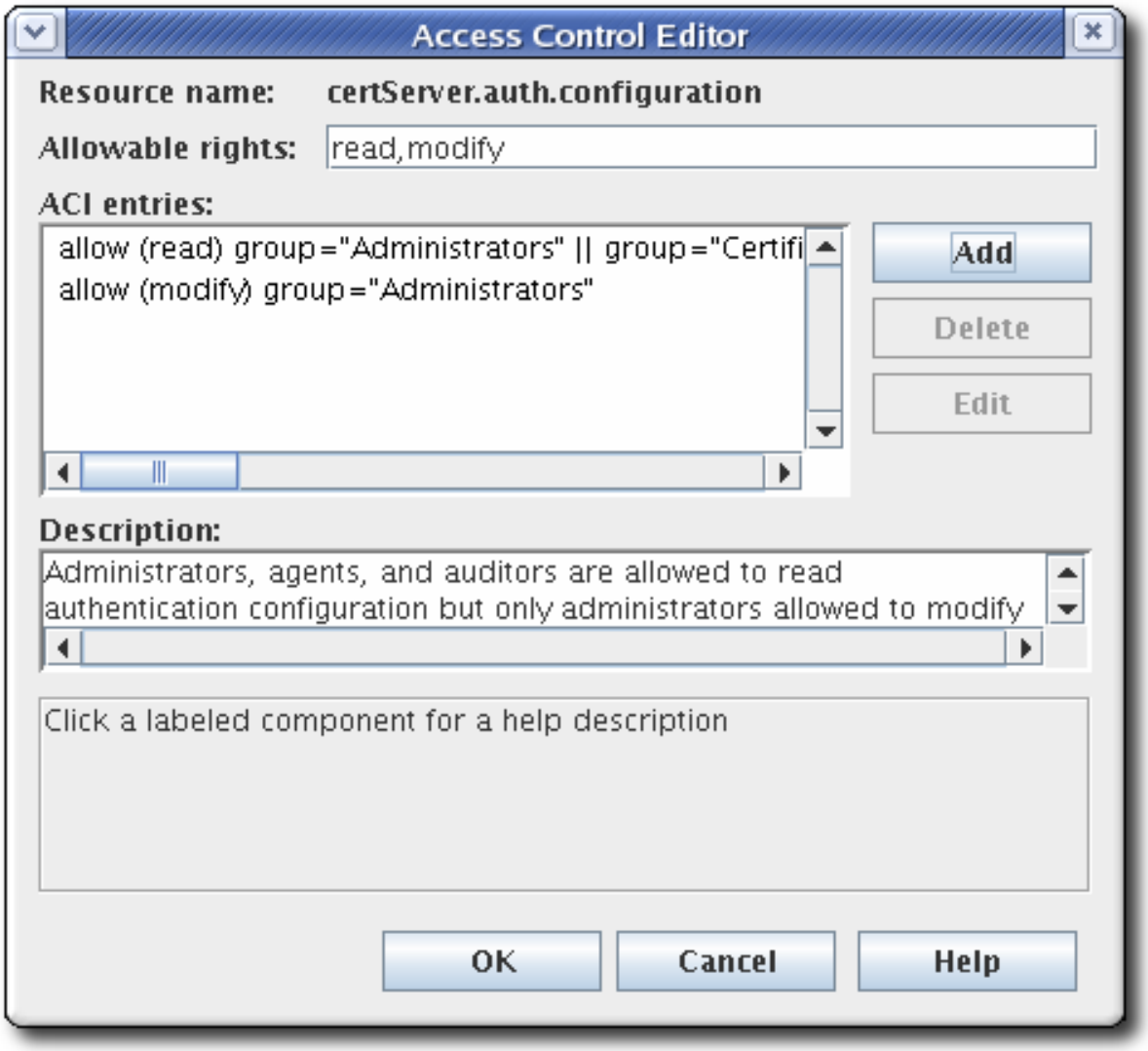
1. 管理コンソールにログインします。

2. 左側のナビゲーションメニューで、**Access Control List** を選択します。



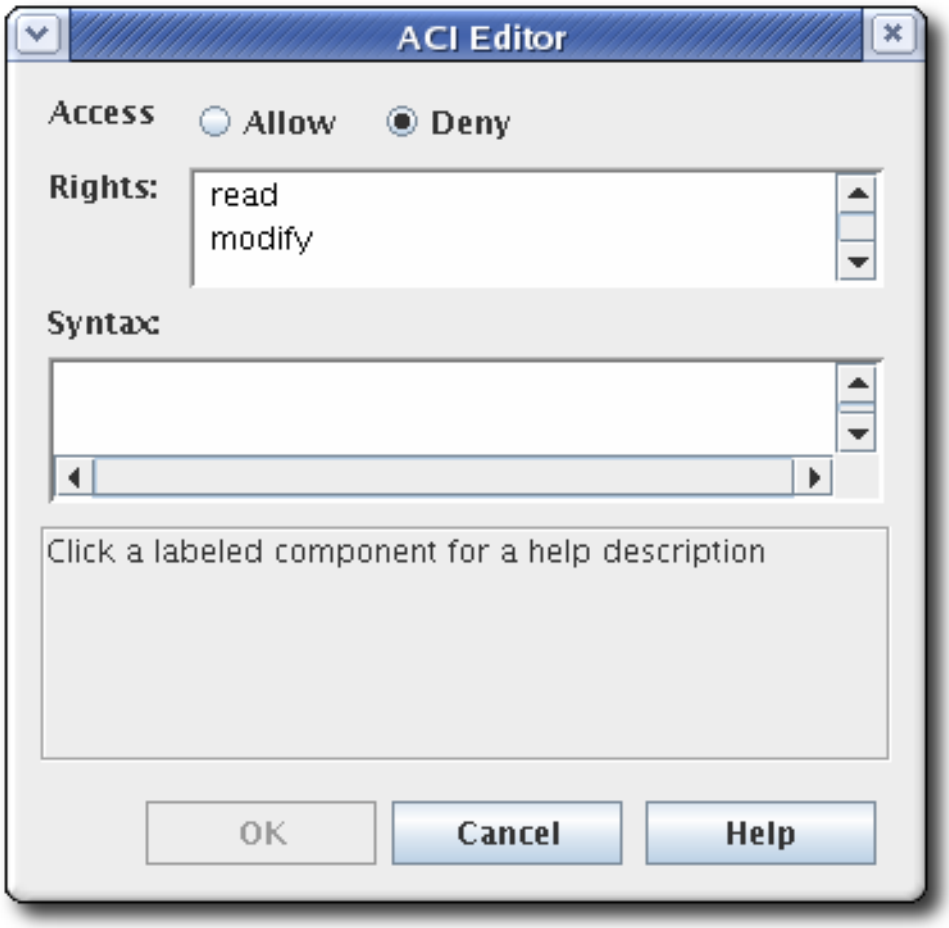
3. リストから編集する ACL を選択し、**Edit** をクリックします。

   アクセス制御エディター ウィンドウで ACL が開きます。



4. ACI を追加するには、**Add** をクリックし、ACI 情報を指定します。

   ACI を編集するには、**ACL Editor** 画面の **ACI entries** テキスト領域で ACI を選択します。**Edit** をクリックします。



a. 指定したグループ、ユーザー、または IP アドレスへの操作を許可または拒否するには、**Access** フィールドから allow または deny ラジオボタンを選択します。アクセスの許可または拒否に関する詳細は、「アクセス制御について」を参照してください。

b. アクセス制御の権限を設定します。オプションは **read** および **modify** です。両方を設定するには、**Ctrl** ボタンまたは **Shift** ボタンを使用します。

c. **Syntax** フィールドでアクセスを許可または拒否されるユーザー、グループ、または IP アドレスを指定します。構文の詳細は、「アクセス制御について」を参照してください。

第13章 サブシステムログの設定

ログの概要については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Certificate System Architecture Overview』 セクションの 『Logs』 セクションを参照してください。

インストール中のログ設定および追加情報については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Configuring Logs』 セクションを参照してください。

13.1. ログの管理

Certificate System サブシステムログファイルは、その特定のサブシステムインスタンス内の操作に関連するイベントを記録します。サブシステムごとに、インストール、アクセス、Web サーバーなどの問題について異なるログが保持されます。

すべてのサブシステムには同様のログ設定、オプション、および管理パスがあります。

13.1.1. コンソールでログの設定

ログはサブシステムコンソールから設定できます。署名付き監査ログやカスタムログなどの特別なログは、コンソールまたは設定ファイルからも作成できます。

1. **Configuration** タブのナビゲーションツリーで **Log** を選択します。
2. **ログイベントリスナー管理** タブには、現在設定されているリスナーが一覧表示されます。

   新しいログインスタンスを作成するには、**Add** をクリックし、**Select Log Event Listener Plug-in Implementation** ウィンドウの一覧からモジュールプラグインを選択します。
3. **Log Event Listener Editor** ウィンドウでフィールドを設定または変更します。さまざまなパラメーターが 表13.1「ログイベントリスナーフィールド」 に一覧表示されます。

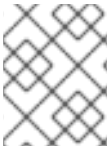
表13.1 ログイベントリスナーフィールド

フィールド	説明
Log Event Listener ID	リスナーを識別する一意の名前を指定します。この名前には、文字 (aA から zZ)、数字 (0 から 9)、アンダースコア (_)、およびハイフン (-) を使用できますが、他の文字やスペースは使用できません。
type	ログファイルのタイプを指定します。**システム** はエラーおよびシステムログを作成します。**トランザクション** は監査ログを記録します。
enabled	ログがアクティブかどうかを設定します。有効にするログのみがイベントを記録します。値は **true** または **false** です。
level	テキストフィールドにログレベルを設定します。このレベルは、フィールドに手動で入力する必要があります。選択メニューはありません。**Debug**、**Information**、**Warning**、**Failure**、**Misconfiguration**、**Catastrophe**、および **Security** を選択できます。

フィールド	説明
fileName	ログファイルへのファイル名を含む完全パスを指定します。サブシステムユーザーには、ファイルへの読み書きパーミッションがなければなりません。
bufferSize	ログのキロバイトサイズ (KB) のバッファーサイズを設定します。バッファーがこのサイズに達すると、バッファーの内容はフラッシュされ、ログファイルにコピーされます。デフォルトのサイズは 512 KB です。
flushInterval	バッファーの内容がフラッシュされてログファイルに追加されるまでの時間を設定します。デフォルトの間隔は 5 秒です。
maxFileSize	ローテーションされる前に可能なログファイルのサイズをキロバイト (KB) 単位で設定できます。このサイズに達すると、ファイルはローテーションファイルにコピーされ、ログファイルが新たに開始されます。デフォルトのサイズは 2000 KB です。
rolloverInterval	アクティブなログファイルをローテートするようにサーバーの頻度を設定します。利用可能なオプションは hourly、daily、weekly、monthly、および yearly です。デフォルトは monthly です。

13.1.2. 監査ログの管理

監査ログには、記録可能なイベントとして設定されたイベントの監査レコードが含まれます。『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の Enable 『and Configuring Signed Audit Logs』 セクションで説明されているように、監査ログ署名オプションを有効にした場合、監査ログは、に属するログ署名証明書で署名されます。サーバー。この証明書は、ログが改ざんされていないことを確認するために監査人が使用できます。



注記

署名付き監査ログは任意です。これを有効にするには、「コンソールでの署名監査ログの設定」 を参照してください。

デフォルトでは、通常の監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/** ディレクトリーと他のタイプのログにありますが、署名済み監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/signedAudit/** に書き込まれます。ログのデフォルトの場所を変更するには、設定を変更してください。

署名された監査ログは、ログ録画システムイベントを作成し、イベントが潜在的なイベント一覧から選択されます。有効にすると、署名された監査ログは、選択したイベントアクティビティーに関するメッセージの詳細セットを記録します。

「コンソールでの署名監査ログの設定」 で説明されているように、設定後に設定の編集や署名証明書の変更も可能です。

13.1.2.1. コンソールでの署名監査ログの設定

署名付き監査ログは、インスタンスの初回作成時にデフォルトで設定されますが、インストール後に設定を編集するか、署名証明書を変更することができます。

注記

署名された監査ログは大きくなる可能性があるため、ファイルシステムに十分なスペースを確保してください。

logSigning パラメーターを **enable** に設定し、ログの署名に使用される証明書のニックネームを指定することにより、ログが署名済み監査ログに設定されます。特別なログ署名証明書は、サブシステムの初回設定時に作成されます。

auditor 権限を持つユーザーのみが、署名済み監査ログにアクセスでき、表示できます。監査担当者は、**AuditVerify** ツールを使用して、署名済み監査ログが改ざんされていないことを確認できます。

署名付き監査ログはサブシステムの設定時に作成され、有効になりますが、監査ログの作成および署名を行うには追加の設定が必要になります。

1. コンソールを開きます。
2. **Configuration** タブのナビゲーションツリーで **Log** を選択します。
3. **ログイベントリスナー管理** タブで、**SignedAudit** エントリーを選択します。
4. **Edit/View** をクリックします。
5. **Log Event Listener Editor** ウィンドウでリセットする必要があるフィールドは 2 つあります。
   - **logSigning** フィールドを **true** に設定して、署名済みロギングを有効にします。

     注記

     よりきめ細かい監査イベントを選択するには、インストール設定時に監査イベントフィルターを設定します。詳細は、『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』の 『Filtering Audit Events』 セクションを参照してください。

   - 監査ログに記録される **イベント** を設定します。付録E *監査イベント* ログ可能なイベントを一覧表示します。ログイベントは、空白のないコンマで区切ります。
6. ログ設定を保存します。

署名付き監査ログを有効にした後、ユーザーを作成し、そのエントリーを監査人グループに割り当てることにより、監査人ユーザーを割り当てます。監査グループのメンバーは、署名された監査ログを表示して検証できる唯一のユーザーです。auditors の設定に関する詳細は、「ユーザーの作成」 を参照してください。

監査担当者は、**AuditVerify** ツールを使用してログを確認できます。このツールの使用方法は、man ページの **AuditVerify(1)** を参照してください。詳細は、「署名監査ログの使用」 を参照します。

13.1.2.2. 監査ロギングエラーの処理

監査ロギング機能が失敗する可能性があるイベントがあるため、イベントをログに書き込むことができません。たとえば、監査ログファイルが含まれるファイルシステムが満杯であったり、ログファイルのファイル権限が誤って変更されると、監査ログのロギングが失敗する可能性があります。

ログ署名が有効で、監査ログが失敗した場合、Certificate System インスタンスは次の方法でシャットダウンします。

- サーブレットが無効になり、新しいリクエストを処理しません。
- 保留中のリクエストと新しいリクエストはすべて強制終了されます。
- サブシステムがシャットダウンしています。

これが発生すると、管理者と監査人はオペレーティングシステム管理者と協力して、ディスク領域またはファイルパーティションの問題を解決する必要があります。IT の問題が解決したら、監査人は最後の監査ログエントリーが署名されていることを確認する必要があります。これが完了すると、管理者は Certificate System を再起動することができます。

13.2. ログの使用

13.2.1. コンソールでログの表示

サブシステムのトラブルシューティングを行うには、サーバーがログ記録したエラーまたは情報メッセージを確認します。ログファイルを調べると、サーバーの操作の多くの側面も監視できます。一部のログファイルは、コンソールで表示できます。ただし、監査ログには、「署名監査ログの使用」 で説明している方法を使用して、Auditor ロールを持つユーザーのみがアクセスできます。

アクティブまたはローテーションされたシステムログの内容を表示するには、次を実行します。

1. コンソールにログインします。
2. **Status** タブを選択します。
3. **Logs** で表示するログを選択します。
4. **Display Options** セクションで表示設定を行います。
   - **Entries**: 表示するエントリーの最大数。この制限に達すると、Certificate System は検索要求に一致するエントリーを返します。ゼロ (0) はメッセージが返されないことを意味します。フィールドが空の場合、サーバーは見つかった数に関係なく、一致するすべてのエントリーを返します。
   - **Source**: ログメッセージが表示される証明書システムコンポーネントまたはサービスを選択します。**All** を選択すると、このファイルにログ記録するすべてのコンポーネントによってログに記録されるメッセージが表示されます。
   - **レベル**: メッセージのフィルターに使用するログレベルを表すメッセージカテゴリーを選択します。
   - **ファイル名**: 表示するログファイルを選択します。現在アクティブなシステムログファイルを表示するには、**Current** を選択します。
5. **Refresh** をクリックします。

   この表には、システムログエントリーが表示されます。エントリーは逆順で、最新のエントリーが一番上に置かれています。パネルの右にあるスクロールバーを使用して、ログエントリーを下方向にスクロールします。

   各エントリーには、以下の情報が表示されます。

   - **ソース**: メッセージをログに記録したコンポーネントまたはリソース
   - **level**: 対応するエントリーの重大度。

- **Date**: エントリーがログに記録された日付。
- **時間**: エントリーがログに記録された時間。
- **詳細**: ログの簡単な説明

6. 完全なエントリーを表示するには、エントリーをダブルクリックしてそのエントリーを選択し、**View** をクリックします。

13.2.2. 署名監査ログの使用

このセクションでは、署名された監査ログを Auditor グループのユーザーが表示および検証する方法を説明します。

13.2.2.1. 監査ログの一覧表示

auditor 権限を持つユーザーは、**pki *subsystem*-audit-file-find** コマンドを使用して、サーバー上の既存の監査ログファイルを一覧表示します。

たとえば、**server.example.com** にホストされる CA の監査ログファイルを一覧表示するには、次のコマンドを実行します。

```
# 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
----------------------------
```

このコマンドは、CA に対して認証するために、**auditor** ディレクトリーに保存されている *auditor* ニックネームのあるクライアント証明書を使用します。コマンドで使用するパラメーターおよび代替の認証方法の詳細は、man ページの pki(1) を参照してください。

13.2.2.2. 監査ログのダウンロード

auditor 権限を持つユーザーとして、**pki *subsystem*-audit-file-retrieve** コマンドを使用して、サーバーから特定の監査ログをダウンロードします。

たとえば、**server.example.com** でホストされる CA から監査ログファイルをダウンロードするには、以下を実行します。

1. 任意で、CA で利用可能なログファイルを一覧表示します。「監査ログの一覧表示」を参照してください。
2. ログファイルをダウンロードします。たとえば、**ca_audit** ファイルをダウンロードします。

```
# pki -U https://server.example.com:8443 -n auditor ca-audit-file-retrieve ca_audit
```

このコマンドは、CA に対して認証するために、**auditor** ディレクトリーに保存されている *auditor* ニックネームのあるクライアント証明書を使用します。コマンドで使用するパラメーターおよび代替の認証方法の詳細は、man ページの pki(1) を参照してください。

ログファイルをダウンロードした後、**grep** ユーティリティーを使用して、特定のログエントリーを検索できます。

```
# grep "\[AuditEvent=ACCESS_SESSION_ESTABLISH\]" log_file
```

13.2.2.3. 署名済み監査ログの確認

監査ログの署名が有効な場合、監査権限を持つユーザーはログを確認することができます。

1. NSS データベースを初期化し、CA 証明書をインポートします。詳細は、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Command-line Initialization』 セクションを参照してください。
2. 監査署名証明書が PKI クライアントデータベースにない場合は、インポートします。
   a. 確認するサブシステムログについて監査署名証明書を検索します。以下に例を示します。

```
# 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. 監査署名証明書を PKI クライアントにインポートします。

```
# pki client-cert-import "CA Audit Signing Certificate" --serial 0x5 --trust ",,P"
-----------------------------------------------------
Imported certificate "CA Audit Signing Certificate"
-----------------------------------------------------
```

3. 監査ログをダウンロードします。「監査ログのダウンロード」を参照してください。
4. 監査ログを確認します。
   a. 検証する監査ログファイルのリストを時系列で含むテキストファイルを作成します。以下に例を示します。

```
# cat > ~/audit.txt << EOF
ca_audit.20170331225716
ca_audit.20170401001030
ca_audit
EOF
```

b. **AuditVerify** ユーティリティーを使用して署名を確認します。以下に例を示します。

```
# AuditVerify -d ~/.dogtag/nssdb/ -n "CA Audit Signing Certificate" \
    -a ~/audit.txt
Verification process complete.
Valid signatures: 10
Invalid signatures: 0
```

AuditVerify の使用方法は、AuditVerify(1) man ページを参照してください。

13.2.3. オペレーティングシステムレベルの監査ログの表示

注記

以下の手順を使用して Operating System レベルの監査ログを表示するには、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『OS レベルの監査ログの有効化』 セクションに従って **auditd** ロギングフレームワークを設定する必要があります。

オペレーティングシステムレベルのアクセスログを表示するには、root として **ausearch** ユーティリティーを使用するか、**sudo** ユーティリティーを使用して特権ユーザーとして使用します。

13.2.3.1. 監査ログ削除イベントの表示

これらのイベントは、(**rhcs_audit_deletion** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_deletion
```

13.2.3.2. シークレットおよび秘密鍵の NSS データベースへのアクセスの表示

これらのイベントは、(**rhcs_audit_nssdb** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_nssdb
```

13.2.3.3. 時間変更イベントの表示

これらのイベントはキー化されるため (**rhcs_audit_time_change**を使用)、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_time_change
```

13.2.3.4. パッケージ更新イベントの表示

これらのイベントは、(**SOFTWARE_UPDATE** タイプの) タイプ付きメッセージであるため、***-m*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -m SOFTWARE_UPDATE
```

13.2.3.5. PKI 設定変更の表示

これらのイベントは、(**rhcs_audit_config** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_config
```

第14章 サブシステム証明書の管理

本章では、証明書の使用の概要を示します。使用できるタイプと形式、HTML エンドエンティティーフォームと Certificate System コンソールを使用して証明書を要求および作成する方法、Certificate System とさまざまなクライアントに証明書をインストールする方法です。さらに、コンソールを介した証明書の管理と、証明書を使用するためのサーバー設定に関する情報があります。

14.1. 必要なサブシステム証明書

各サブシステムには、操作を実行するためにサブシステムインスタンスに発行する必要のある証明書の定義されたセットがあります。Certificate Manager の設定中に設定される証明書の内容には特定の詳細があり、証明書の種類に応じて、制約、設定、および属性についてさまざまな考慮事項があります。証明書の形式の計画については、『Red Hat Certificate System 9 Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Types of Certificates』 セクションで説明されています。

14.1.1. 証明書マネージャー証明書

Certificate Manager がインストールされると、CA 署名証明書、TLS サーバー証明書、および OCSP 署名証明書の鍵および要求が生成されます。この証明書は、設定を完了する前に作成されます。

CA 証明書要求は、CA への自己署名リクエストとして送信されます。次に、証明書を発行して自己署名ルート CA の作成を終了するか、サードパーティーのパブリック CA または別の Certificate System CA に送信されます。外部 CA が証明書を返すと、証明書がインストールされ、下位 CA のインストールが完了します。

- 「CA 署名キーペアおよび証明書」
- 「OCSP 署名キーペアおよび証明書」
- 「サブシステム証明書」
- 「TLS サーバーキーペアおよび証明書」
- 「Audit ログ署名キーペアおよび証明書」

14.1.1.1. CA 署名キーペアおよび証明書

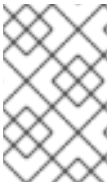
すべての Certificate Manager には、Certificate Manager が発行する証明書と CRL に署名するために使用する秘密鍵に対応する公開鍵を持つ CA 署名証明書があります。この証明書は、Certificate Manager のインストール時に作成され、インストールされます。証明書のデフォルトのニックネームは **caSigningCert cert-instance_ID CA** です。ここで、**instance_ID** は Certificate Manager インスタンスを識別します。証明書のデフォルトの有効期間は 5 年間です。

CA 署名証明書のサブジェクト名は、インストール時に設定された CA の名前を反映します。Certificate Manager によって署名または発行されたすべての証明書には、証明書の発行者を識別するためにこの名前が含まれています。

Certificate Manager のステータスがルートまたは下位 CA として評価されるかどうかは、その CA 署名証明書が自己署名の証明書であるか、または別の CA により署名されているかにより決まります。これは、証明書のサブジェクト名に影響します。

- Certificate Manager がルート CA の場合、その CA 署名証明書は自己署名の証明書です。つまり、証明書のサブジェクト名と発行者名は同じです。

- Certificate Manager が下位 CA である場合、その CA 署名証明書は別の CA、通常は CA 階層の上位レベル (ルート CA である場合とそうでない場合があります) によって署名されます。Certificate Manager を使用して証明書を発行する前に、ルート CA の署名証明書を個別のクライアントおよびサーバーにインポートする必要があります。



注記

CA 名を変更 **できない** か、以前に発行された証明書がすべて無効化されている。同様に、新しい鍵ペアで CA 署名証明書を再発行し、古い鍵ペアで署名された証明書をすべて無効にします。

14.1.1.2. OCSP 署名キーペアおよび証明書

OCSP 署名証明書のサブジェクト名は **cn=OCSP cert-**instance_ID **CA** の形式であり、OCSP レスポンスの署名に必要な拡張機能 (**OCSPSigning** および **OCSPNoCheck** など) が含まれます。

OCSP 署名証明書のデフォルトのニックネームは **ocspSigningCert cert-**instance_ID で、**instance_ID CA** は証明書マネージャーインスタンスを識別します。

OCSP 署名証明書の公開鍵に対応する OCSP 秘密鍵は、証明書失効リストのステータスをクエリーするときに Certificate Manager が OCSP 準拠のクライアントに署名するために使用されます。

14.1.1.3. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー TLS 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。セキュリティードメイン CA 自体の場合は、そのサブシステム証明書自体によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-**instance_ID です。

14.1.1.4. TLS サーバーキーペアおよび証明書

すべての Certificate Manager には、Certificate Manager のインストール時に最初に生成された TLS サーバー証明書が少なくとも 1 つあります。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。**instance_ID** は証明書マネージャーインスタンスを識別します。

デフォルトでは、認証に Certificate Manager は TLS サーバー証明書を使用します。ただし、エンドエンティティーサービスインターフェイスとエージェントサービスインターフェイスへの認証に個別のサーバー証明書を使用するように証明書マネージャーを設定するなど、さまざまな操作に使用する追加のサーバー証明書を要求できます。

Certificate Manager が公開ディレクトリーとの TLS 対応通信用に設定されている場合、デフォルトではクライアント認証に TLS サーバー証明書を使用します。Certificate Manager は、TLS クライアント認証に別の証明書を使用するように設定することもできます。

14.1.1.5. Audit ログ署名キーペアおよび証明書

CA は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

14.1.2. オンライン証明書ステータスマネージャーの証明書

Online Certificate Status Manager を最初に設定すると、必要なすべての証明書のキーが作成され、OCSP 署名、TLS サーバー、監査ログ署名、およびサブシステム証明書の証明書要求が行われます。これらの証明書要求は CA (Certificate System CA またはサードパーティー CA のいずれか) に送信され、設定プロセスを完了するには Online Certificate Status Manager データベースにインストールする必要があります。

- 「TLS サーバーキーペアおよび証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」
- 「オンライン証明書ステータスマネージャー証明書の認識」

14.1.2.1. OCSP 署名キーペアおよび証明書

すべての Online Certificate Status Manager には、証明書である OCSP 署名証明書があります。これには、Online Certificate Status Manager が OCSP 応答に署名するために使用する秘密鍵に対応する公開鍵があります。Online Certificate Status Manager の署名は、Online Certificate Status Manager がリクエストを処理している永続的な証明を提供します。この証明書は、Online Certificate Status Manager が設定されている場合に生成されます。証明書のデフォルトのニックネームは **ocspSigningCert cert-instance_ID** で、**instance_ID OSCP** は Online Certificate Status Manager インスタンス名です。

14.1.2.2. TLS サーバーキーペアおよび証明書

すべての Online Certificate Status Manager には、Online Certificate Status Manager の設定時に生成された TLS サーバー証明書が少なくとも 1 つあります。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。ここで、**instance_ID** は Online Certificate Status Manager インスタンス名を識別します。

Online Certificate Status Manager は、Online Certificate Status Manager エージェントサービスページでサーバー側の認証にサーバー証明書を使用します。

Online Certificate Status Manager は認証目的で単一のサーバー証明書を使用します。追加のサーバー証明書をインストールして、さまざまな目的で使用できます。

14.1.2.3. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー TLS 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-instance_ID** です。

14.1.2.4. Audit ログ署名キーペアおよび証明書

OCSP は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

注記

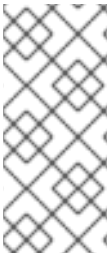
その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

14.1.2.5. オンライン証明書ステータスマネージャー証明書の認識

Online Certificate Status Manager の TLS サーバー証明書に署名した CA によっては、Certificate Manager が認識する証明書および発行する CA を取得しないといけない場合があります。

- Online Certificate Status Manager のサーバー証明書が CRL を公開している CA によって署名されている場合は、何もする必要はありません。
- Online Certificate Status Manager のサーバー証明書が、下位の Certificate Manager の証明書に署名したのと同じルート CA によって署名されている場合、ルート CA は、下位の Certificate Manager の証明書データベースで信頼できる CA としてマークする必要があります。
- Online Certificate Status Manager の TLS サーバー証明書が別のルート CA によって署名されている場合は、ルート CA 証明書を下位の Certificate Manager の証明書データベースにインポートし、信頼できる CA としてマークする必要があります。

Online Certificate Status Manager のサーバー証明書が、選択したセキュリティードメインの CA によって署名されている場合は、Online Certificate Status Manager の設定時に証明書チェーンがインポートされ、マークされます。他の設定は必要ありません。ただし、サーバー証明書が外部 CA で署名されている場合は、設定を完了するために証明書チェーンをインポートする必要があります。



注記

セキュリティードメイン内のすべての CA が、設定時に OCSP Manager によって自動的に信頼されるわけではありません。ただし、CA パネルで設定された CA の証明書チェーン内のすべての CA は、OCSP Manager によって自動的に信頼されます。セキュリティードメイン内にあるが証明書チェーンにはない他の CA は、手動で追加する必要があります。

14.1.3. キーリカバリー認証局の証明書

KRA は、以下のキーペアと証明書を使用します。

- 「トランスポートキーペアおよび証明書」
- 「ストレージキーペア」
- 「TLS サーバー証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

14.1.3.1. トランスポートキーペアおよび証明書

すべての KRA にはトランスポート証明書があります。トランスポート証明書の生成に使用されるキーペアの公開キーは、アーカイブのために KRA に送信される前に、エンドエンティティーの秘密暗号化

キーを暗号化するためにクライアントソフトウェアによって使用されます。デュアルキーペアを生成できるクライアントのみがトランスポート証明書を使用します。

14.1.3.2. ストレージキーペア

すべての KRA にはストレージキーペアがあります。 KRA は、このキーペアの公開コンポーネントを使用して、キーをアーカイブするときに秘密暗号化キーを暗号化 (またはラップ) します。プライベートコンポーネントを使用して、リカバリー中にアーカイブされたキーを復号化 (またはアンラップ) します。このキーペアの使用方法に関する詳細は、*4章 キーアーカイブおよびリカバリーの設定* を参照してください。

ストレージキーで暗号化したキーは、承認されたキーリカバリーエージェントによりのみ取得できます。

14.1.3.3. TLS サーバー証明書

すべての Certificate System KRA には、少なくとも 1 つの TLS サーバー証明書があります。KRA が設定されると、最初の TLS サーバー証明書が生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-**instance_ID です。**instance_id** は KRA インスタンスを識別します。

KRA の TLS サーバー証明書は、証明書要求が送信される CA により発行されました。これは Certificate System CA またはサードパーティー CA です。発行者名を表示するには、KRA コンソールの **System Keys and Certificates** オプションで証明書の詳細を開きます。

KRA は、KRA エージェントサービスインターフェイスへのサーバー側認証に TLS サーバー証明書を使用します。デフォルトでは、キー回復機関は認証に 1 つの TLS サーバー証明書を使用します。ただし、追加の TLS サーバー証明書を要求して、KRA 用にインストールすることができます。

14.1.3.4. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー TLS 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-**instance_ID です。

14.1.3.5. Audit ログ署名キーペアおよび証明書

KRA は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

14.1.4. TKS 証明書

TKS には 3 つの証明書があります。TLS サーバーとサブシステムの証明書は、標準操作に使用されます。監査ログの保護には、追加の署名証明書が使用されます。

- 「TLS サーバー証明書」

- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

14.1.4.1. TLS サーバー証明書

すべての Certificate System TKS には、少なくとも 1 つの TLS サーバー証明書があります。最初の TLS サーバー証明書は、TKS の設定時に生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。

14.1.4.2. サブシステム証明書

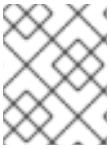
セキュリティードメインのすべてのメンバーには、サーバー TLS 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-instance_ID** です。

14.1.4.3. Audit ログ署名キーペアおよび証明書

TKS は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。



注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

14.1.5. TPS 証明書

TPS は、サーバー証明書、サブシステム証明書、監査ログ署名証明書の 3 つの証明書のみを使用します。

- 「TLS サーバー証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

14.1.5.1. TLS サーバー証明書

すべての Certificate System TKS には、少なくとも 1 つの TLS サーバー証明書があります。最初の TLS サーバー証明書は、TPS が設定されるときに生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。

14.1.5.2. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー TLS 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-instance_ID** です。

14.1.5.3. Audit ログ署名キーペアおよび証明書

TPS は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

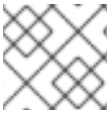
14.1.6. サブシステム証明書のキータイプについて

新規インスタンスの作成時には、**pkispawn** ユーティリティーに渡される設定ファイルでキーのタイプとキーサイズを指定できます。

例14.1 CA のキータイプ関連の設定パラメーター

以下は、例の値を含む主要なタイプ関連のパラメーターです。これらのパラメーターは、新規 CA の作成時に **pkispawn** に渡される設定ファイルで設定できます。

```
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
```



注記

サンプルの値は CA 用です。他のサブシステムには異なるパラメーターが必要です。

詳細は、以下を参照してください。

- 『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Understanding the **pkispawn** Utility』 セクション。
- パラメーターおよび例の説明の pki_default.cfg(5) man ページ。

14.1.7. HSM を使用したサブシステム証明書の保存

デフォルトでは、鍵と証明書は、**/var/lib/pki/*instance_name*/alias/** ディレクトリーのローカル管理のデータベースである **key3. db および cert8**.db に保存されます。ただし、Red Hat Certificate System は、ハードウェアセキュリティーモジュール (HSM) もサポートします。この外部デバイスは、ネットワーク上にある集中的な場所に鍵と証明書を保存できます。HSM を使用すると、インスタンスのキーと証明書に簡単にアクセスできるため、クローン作成などの一部の機能が簡単になります。

HSM を使用して証明書を保存する場合、HSM 名は証明書のニックネームに付加され、サブシステム設定にフルネームが使用されます。以下に例を示します。

```
serverCert="nethsm:Server-Cert cert-instance_ID
```

注記

1 つの HSM を使用して、複数のホストにインストールする可能性がある複数のサブシステムインスタンスの証明書および鍵を保存することができます。HSM を使用する場合、サブシステムの証明書ニックネームは HSM で管理される各サブシステムインスタンスに対して一意である必要があります。

証明書システムは、nCipher netHSM と Chrysalis LunaSA の 2 種類の HSM に対応します。

14.2. サブシステム証明書の更新

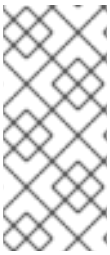
サブシステム証明書の更新の詳細については、「更新について」 を参照してください。

サブシステム証明書を更新するプロセスは、ユーザー証明書を更新するプロセスと同じです。

システム証明書を更新するには、**HttpClient** ユーティリティーを使用する際に、対応する登録プロファイルを使用して要求を送信します。異なるシステム証明書プロファイルの詳細は、「システム証明書およびサーバー証明書の取得」 を参照してください。

14.2.1. certutil を使用した証明書の更新

certutil は、証明書データベースの既存のキーペアを使用して証明書要求を生成するのに使用できます。次に、新しい証明書要求を CMC 要求に変換して、CA に送信できます。詳細は、「**certutil** を使用した CSR の作成」 を参照してください。



注記

暗号化および署名証明書は単一のステップで作成されます。ただし、更新プロセスは一度に 1 つの証明書のみを更新します。

証明書ペアで両方の証明書を更新するには、各証明書を個別に更新する必要があります。

1. トークンデータベースのパスワードを取得します。

```
cat /var/lib/pki/instance_name/conf/password.conf

internal=263163888660
```

2. 証明書を更新しているインスタンスの証明書データベースディレクトリーを開きます。

```
cd /var/lib/pki/instance_name/alias
```

3. 更新する証明書のキーとニックネームを一覧表示します。証明書を更新するには、生成に使用されるキーペアと、新しい証明書に指定されたサブジェクト名は、古い証明書の証明書と同じである必要があります。

```
# 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. **alias** ディレクトリーをバックアップとしてコピーし、証明書データベースから元の証明書を削除します。以下に例を示します。

```
certutil -D -n "ServerCert cert-example"  -d .
```

5. 既存の証明書の値にオプションを設定して **certutil** コマンドを実行します。

```
certutil -d . -R -k "NSS Certificate DB:cert-pki-tomcat CA" -s "cn=CA Authority,o=Example Domain" -a -o example.req2.txt
```

新しい証明書とキーのペアの生成と証明書の更新の違いは、**-k** オプションの値です。新しいリクエストとキーのペアを生成するには、**-k** はキータイプを設定してから **-g** で使用します。これは、ビット長を設定します。更新要求では、**-k** オプションは証明書のニックネームを使用してセキュリティーデータベースに保存された既存のキーペアにアクセスします。

パラメーターの詳細は、certutil(1) の man ページを参照してください。

6. 証明書要求を送信し、それを取得してインストールします。

14.2.2. 期限切れの Certificate System サーバー証明書の更新

Certificate System は、PKI サーバーの実行中にサーバー証明書をオンラインで自動的に更新しません。一般に、管理者はシステム証明書が期限切れになる前に更新する必要があります。ただし、システム証明書の期限が切れると、証明書システムが起動できません。

期限切れ後にシステム証明書を更新するために、一時的なサーバー証明書をセットアップできます。

1. システム証明書の有効期限が切れている場合は、以下を行います。

   a. 一時証明書を作成します。

   ```
   # pki-server cert-create sslserver --temp
   ```

   b. Certificate System の Network Security Services (NSS) データベースに一時証明書をインポートします。

   ```
   # pki-server cert-import sslserver
   ```

   c. Certificate System を開始します。

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

2. 証明書を表示し、期限切れのシステム証明書の ID をメモします。

```
# pki-server cert-find
```

3. 新しい永続的な証明書を作成します。

```
# pki-server cert-create certificate_ID
```

4. Certificate System を停止します。

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

5. 新しい証明書をインポートして、期限切れの証明書を置き換えます。

```
# pki-server cert-import certificate_ID
```

6. Certificate System を開始します。

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

14.3. サブシステム証明書の名前の変更

証明書の更新方法の 1 つに、新しい証明書に置き換える方法があります。つまり、新しい証明書が新しい鍵で生成されます。通常、新しい証明書をデータベースに追加し、古い証明書を削除して、シンプルな 1 対 1 のスワップに追加できます。これは、個別のサブシステムサーバーがニックネームに基づいて証明書を識別するためです。証明書のニックネームが同じままであれば、サブジェクト名、シリアル番号、キーなどの他の要素が異なる場合でも、サーバーは必要な証明書を見つけることができます。

以下の表では、サブシステムの証明書の各設定パラメーターを一覧表示します。

- 表14.1「CA Certificate Nickname パラメーター」
- 表14.2「KRA Certificate ニックネームパラメーター」
- 表14.3「OCSP Certificate ニックネームパラメーター」
- 表14.4「TKS Certificate ニックネームパラメーター」
- 表14.5「TPS ニックネームパラメーター」

表14.1 CA Certificate Nickname パラメーター

CA 署名証明書	• ca.cert.signing.nickname • ca.signing.cacertnickname • ca.signing.certnickname • ca.signing.nickname • cloning.signing.nickname
OCSP 署名証明書	• ca.ocsp_signing.cacertnickname • ca.ocsp_signing.certnickname • ca.cert.ocsp_signing.nickname • ca.ocsp_signing.nickname • cloning.ocsp_signing.nickname
サブシステム証明書	• ca.cert.subsystem.nickname • ca.subsystem.nickname • cloning.subsystem.nickname • pkiremove.cert.subsystem.nickname
サーバー証明書	• ca.sslserver.nickname • ca.cert.sslserver.nickname
監査署名証明書	• ca.audit_signing.nickname • ca.cert.audit_signing.nickname • cloning.audit_signing.nickname

表14.2 KRA Certificate ニックネームパラメーター

トランスポート証明書	• cloning.transport.nickname • kra.cert.transport.nickname • kra.transport.nickname • tks.kra_transport_cert_nickname このパラメーターは TKS 設定ファイルにあることに注意してください。TKS 証明書がすべて同じままであっても、 **KRA** トランスポート証明書のニックネームが変更された場合は、TKS 設定でこれを変更する必要があります。
ストレージ証明書	• cloning.storage.nickname • kra.storage.nickname • kra.cert.storage.nickname
サーバー証明書	• kra.cert.sslserver.nickname • kra.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • kra.cert.subsystem.nickname • kra.subsystem.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• cloning.audit_signing.nickname • kra.cert.audit_signing.nickname • kra.audit_signing.nickname

表14.3 OCSP Certificate ニックネームパラメーター

OCSP 署名証明書	• cloning.signing.nickname • ocsp.signing.certnickname • ocsp.signing.cacertnickname • ocsp.signing.nickname

サーバー証明書	• ocsp.cert.sslserver.nickname • ocsp.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • ocsp.subsystem.nickname • ocsp.cert.subsystem.nickname • pkiremove.cert.subsystem
監査ログ署名証明書	• cloning.audit_signing.nickname • ocsp.audit_signing.nickname • ocsp.cert.audit_signing.nickname

表14.4 TKS Certificate ニックネームパラメーター

KRA 転送証明書[a]	• tks.kra_transport_cert_nickname
サーバー証明書	• tks.cert.sslserver.nickname • tks.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • tks.cert.subsystem.nickname • tks.subsystem.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• cloning.audit_signing.nickname • tks.audit_signing.nickname • tks.cert.audit_signing.nickname

[a] TKS 証明書がすべて同じままであっても、 **KRA** トランスポート証明書のニックネームが変更された場合は、TKS 設定でこれを変更する必要があります。

表14.5 TPS ニックネームパラメーター

サーバー証明書	• tps.cert.sslserver.nickname
サブシステム証明書	• tps.cert.subsystem.nickname • selftests.plugin.TPSValidity.nickname • selftests.plugin.TPSPresence.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• tps.cert.audit_signing.nickname

14.4. 証明書データベースの管理

各 Certificate System インスタンスには、内部トークンで維持される証明書データベースがあります。このデータベースには、Certificate System インスタンスにインストールされているサブシステムに属する証明書と、サブシステムが受信する証明書の検証に使用する各種 CA 証明書が含まれます。

外部トークンを使用してキーペアを生成および保存する場合でも、Certificate System は常に、信頼できる CA 証明書と信頼できない CA 証明書のリストを内部トークンに保持します。

本セクションでは、Certificate System ウィンドウを使用して、証明書データベースの内容を表示する方法、不要な証明書を削除する方法、およびデータベースにインストールされている CA 証明書の信頼設定を変更する方法を説明します。データベースに証明書を追加する方法は、「証明書システムデータベースでの証明書のインストール」 を参照してください。

注記

Certificate System コマンドラインユーティリティー **certutil** は、信頼設定を編集し、証明書を追加または削除して、証明書データベースを管理するのに使用できます。このツールの詳細は、http://www.mozilla.org/projects/security/pki/nss/tools/ を参照してください。

管理者は、証明書データベースの内容を定期的にチェックして、不要な CA 証明書が含まれていないことを確認する必要があります。たとえば、データベースに PKI セットアップ内で信頼されるべきではない CA 証明書が含まれている場合は、それらを削除します。

14.4.1. 証明書システムデータベースでの証明書のインストール

サブシステムに対して新しいサーバー証明書が発行された場合は、そのサブシステムデータベースにインストールする必要があります。さらに、ユーザー証明書およびエージェント証明書をサブシステムデータベースにインストールする必要があります。証明書が外部 CA により発行される場合は、通常、対応する CA 証明書または証明書チェーンをインストールする必要があります。

証明書は、Console の Certificate Setup Wizard を使用するか、または **certutil** ユーティリティーを使用してサブシステム証明書データベースにインストールできます。

- 「コンソールを使用した証明書のインストール」
- 「certutil を使用した証明書のインストール」

- 「CA 証明書のチェーンについて」

14.4.1.1. コンソールを使用した証明書のインストール

Certificate Setup Wizard は、Certificate System インスタンスが使用する内部トークンまたは外部トークンのいずれかに以下の証明書をインストールまたはインポートできます。

- Certificate System サブシステムが使用する証明書のいずれか
- 外部 CA またはその他の Certificate System CA からの信頼済み CA 証明書
- 証明書チェーン

証明書チェーンには、証明書のコレクション (サブジェクト証明書、信頼されたルート CA 証明書、およびサブジェクト証明書を信頼されたルートにリンクするために必要な中間 CA 証明書) が含まれます。ただし、ウィザードがインポートする証明書チェーンには、CA 証明書のみを含める必要があります。どの証明書もユーザー証明書にすることはできません。

証明書チェーンでは、チェーンの各証明書は個別の DER でエンコードされたオブジェクトとしてエンコードされます。ウィザードが証明書チェーンをインポートすると、これらのオブジェクトが次々にインポートされ、チェーンの最後の証明書 (ルート CA 証明書である場合とそうでない場合があります) までインポートされます。チェーン内の証明書のいずれかがローカル証明書データベースにすでにインストールされている場合、ウィザードは既存の証明書をチェーン内の証明書に置き換えます。チェーンに中間 CA 証明書が含まれる場合、ウィザードは *信頼されていない* CA 証明書として証明書データベースに追加します。

サブシステムコンソールは、同じウィザードを使用して証明書と証明書チェーンをインストールします。ローカルセキュリティーデータベースに証明書をインストールするには、以下の手順を実施します。

1. コンソールを開きます。

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

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. サブシステムのタイプと証明書のタイプに応じて、証明書をインストールできる 2 つのタブがあります。

   - **CA 証明書** タブは、CA 証明書および証明書チェーンのインストールに使用されます。Certificate Manager の場合、このタブはサードパーティーの CA 証明書またはその他の Certificate System CA 証明書に使用されます。すべてのローカル CA 証明書は、**Local Certificates** タブにインストールされます。その他のサブシステムでは、すべての CA 証明書およびチェーンがこのタブでインストールされます。
   - **Local Certificates** タブには、すべてのサーバー証明書、サブシステム証明書、および OCSP 署名や KRA トランスポートなどのローカル証明書がインストールされます。

   該当するタブを選択します。
4. **Local Certificates** タブで証明書をインストールするには、**Add/Renew** をクリックします。**CA Certificates** タブに証明書をインストールするには、**Add** をクリックします。いずれも Certificate Setup Wizard を開きます。

a. ウィザードが開いたら、**Install a certificate** ラジオボタンを選択し、**Next** をクリックします。

b. インストールする証明書のタイプを選択します。ドロップダウンメニューのオプションは、サブシステムのタイプに応じて、証明書の作成に使用できるオプションと同じですが、クロスペア証明書をインストールするための追加オプションがあります。

c. **-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** を含む証明書の本文にテキストエリアに貼り付けるか、絶対ファイルの場所を指定します。これはローカルファイルでなければなりません。

証明書は以下のようになります。

```
-----BEGIN CERTIFICATE-----
MIICKzCCAZSgAwIBAgIBAzANgkqkiG9w0BAQQFADA3MQswCQYDVQQGEw
JVUzERMA8GA1UEChMITmV0c2NhcGUxFTATBgNVBAsTDFN1cHJpeWEncy
BDQTAeFw05NzEwMTgwMTM2MjVaFw05OTEwMTgwMTM2MjVaMEgxCzAJBg
NVBAYTAlVTMREwDwYDVQQKEwhOZXRzY2FwZTENMAsGA1UECxMEUHawcz
EXMBUGA1UEAxMOU3VwcmI5YSBTaGV0dHkwgZ8wDQYJKoZIhdfNAQEBBQ
ADgY0AMIGJAoGBAMr6eZiPGfjX3uRJgEjmKiqG7SdATYzBcABu1AVyd7
chRFOGD3wNktbf6hRo6EAmM5R1Askzf8AW7LiQZBcrXpc0k4du+2j6xJ
u2MPm8WKuMOTuvzpo+SGXeImHVChEqooCwfdiZywyZNmgaMa2MS6pUkf
QVAgMBAAGjNjA0MBEGCWCGSAGG+EIBAQQEAwIAgD
-----END CERTIFICATE-----
```

5. ウィザードに、証明書の詳細が表示されます。指紋を確認して、これが正しい証明書であることを確認するか、**Back** ボタンをクリックして戻って別のボタンを送信します。証明書のニックネームを指定します。

このウィザードは、証明書をインストールします。

6. 証明書に署名した CA はサブシステムによって信頼される必要があります。この CA の証明書がサブシステムの証明書データベース (内部または外部) に存在し、信頼されていることを確認してください。

CA 証明書がリストされていない場合は、信頼できる CA として証明書を証明書データベースに追加します。CA の証明書がリストされているが信頼されていない場合は、「CA 証明書の信頼設定の変更」 に示すように、信頼設定を信頼済みに変更します。

Certificate System 証明書データベースに保存されていない CA によって発行された証明書をインストールする場合は、その CA の証明書チェーンをデータベースに追加します。CA チェーンをデータベースに追加するには、CA チェーンをテキストファイルにコピーし、ウィザードを再起動して、CA チェーンをインストールします。

14.4.1.2. certutil を使用した証明書のインストール

certutil を使用して Certificate System インスタンスのセキュリティーデータベースにサブシステム証明書をインストールするには、以下を行います。

1. サブシステムのセキュリティーデータベースディレクトリーを開きます。

```
cd /var/lib/pki/instance_name/alias
```

2. **-A** を指定して **certutil** コマンドを実行して、証明書と、CA が発行した証明書が含まれるファイルを示す **-i** を追加しします。

-

```
certutil -A -n cert-name -t trustargs
  -d . -a -i certificate_file
```

注記

Certificate System インスタンスの証明書およびキーが HSM に保存されている場合は、**-h** オプションを使用してトークン名を指定します。

以下に例を示します。

```
certutil -A -n "ServerCert cert-instance_name" -t u,u,u -d . -a -i /tmp/example.cert
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

14.4.1.3. CA 証明書のチェーンについて

証明書をサポートするクライアントまたはサーバーソフトウェアは、証明書データベースに信頼できる CA 証明書のコレクションを保持します。これらの CA 証明書は、ソフトウェアが検証できるその他の証明書を決定します。最も単純なケースでは、ソフトウェアは、証明書を持っている CA の 1 つによって発行された証明書のみを検証できます。信頼できる CA 証明書を CA 証明書のチェーンの一部にすることもできます。各証明書は、証明書階層の上位にある CA によって発行されます。

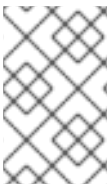
チェーンの最初の証明書は、コンテキスト固有の方法で処理されます。コンテキスト固有の方法は、インポート方法によって異なります。Mozilla Firefox の場合、この処理は、ダウンロードされるオブジェクトで使用される MIME コンテンツタイプによって異なります。Red Hat サーバーでは、サーバー管理インターフェイスで選択したオプションによって異なります。

後続の証明書はすべて同じ扱われます。証明書の Netscape Certificate Type 証明書拡張子に TLS-CA ビットが含まれていて、ローカル証明書データベースにまだ存在しない場合、証明書は信頼できない CA として追加されます。チェーンのどこかに信頼できる CA が存在する限り、証明書チェーンの検証に使用できます。

14.4.2. データベースコンテンツの表示

サブシステム証明書データベースに保存されている証明書 **cert8.db** は、サブシステム管理コンソールから表示できます。または、**certutil** ユーティリティーを使用して一覧表示できます。**certutil** は、TPS サブシステムは管理コンソールを使用しないため、TPS 証明書を表示するのに使用する必要があります。

- 「コンソールを使用したデータベースコンテンツの表示」
- 「certutil を使用したデータベースコンテンツの表示」



注記

cert8.db データベースにリストされている証明書は、サブシステム操作に使用されるサブシステム証明書です。ユーザー証明書は、LDAP 内部データベースのユーザーエントリーと共に保存されます。

14.4.2.1. コンソールを使用したデータベースコンテンツの表示

管理コンソールを使用してデータベースの内容を表示するには、以下を行います。

1. サブシステムコンソールを開きます。

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

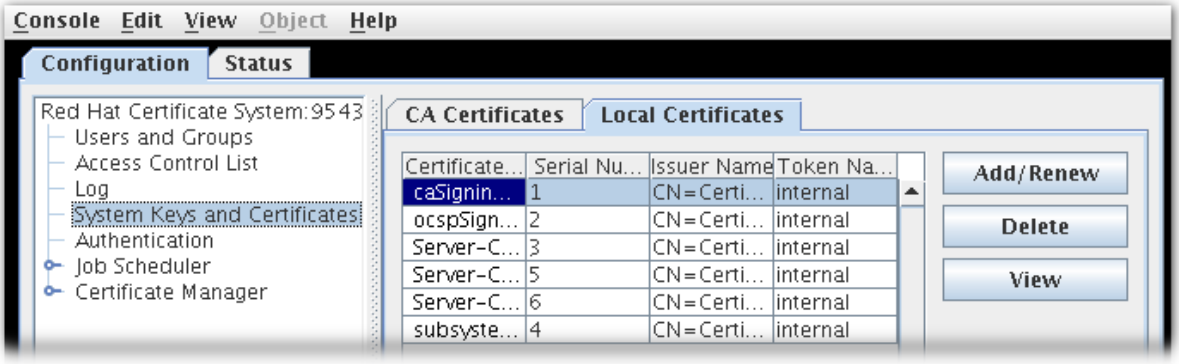
2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。

3. **CA Certificates** および **Local Certificates** の 2 つのタブがあります。ここには、さまざまな種類の証明書が一覧表示されます。

   - **CA 証明書** には、Entrust や Verisign などのサードパーティー CA によって発行された証明書や、外部の Certificate System Certificate Manager など、対応する秘密鍵の資料が利用できない CA 証明書が一覧表示されます。

   - **ローカル証明書** には、KRA トランスポート証明書や OCSP 署名証明書など、Certificate System サブシステムインスタンスによって保持されている証明書が一覧表示されます。

   図14.1 Certificate Database タブ



4. **Certificate Database Management** テーブルには、サブシステムにインストールされている証明書が一覧表示されます。証明書ごとに、以下の情報が提供されます。

   - **Certificate Name**

   - **Serial Number**

   - **Issuer Names**。この証明書の発行者の共通名 (**cn**) です。

   - **Token Name** (証明書を保持する暗号化トークンの名前)。データベースに保存されている証明書の場合、これは **internal** になります。

証明書に関する詳細情報を表示するには、証明書を選択して **View** をクリックします。これにより、証明書のシリアル番号、有効期間、サブジェクト名、発行者名、および証明書のフィンガープリントを表示するウィンドウが開きます。

14.4.2.2. certutil を使用したデータベースコンテンツの表示

certutil を使用してサブシステムデータベース内の証明書を表示するには、インスタンスの証明書データベースディレクトリーを開き、**-L** オプションを付けて **certutil** をインストールします。以下に例を示します。

```
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
```

certutil を使用してサブシステムデータベースに保存されている鍵を表示するには、**-K** オプションを指定して **certutil** を実行します。以下に例を示します。

```
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
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

14.4.3. データベースからの証明書の削除

不要な証明書を削除すると、証明書データベースのサイズが減少します。

注記

証明書データベースから CA 証明書を削除するときは、*中間 CA 証明書* を削除しないように注意してください。これにより、サブシステムが信頼できる CA 証明書にチェーンアップするのに役立ちます。不明な場合は、データベースの証明書を *信頼されていない* CA 証明書として残します。「CA 証明書の信頼設定の変更」 を参照してください。

- 「コンソールを使用した証明書の削除」
- 「certutil を使用した証明書の削除」

14.4.3.1. コンソールを使用した証明書の削除

コンソールから証明書を削除するには、以下の手順を実施します。

1. サブシステムコンソールを開きます。

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

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. 削除する証明書を選択して、**Delete** をクリックします。
4. プロンプトが表示されたら、削除を確定します。

14.4.3.2. certutil を使用した証明書の削除

certutil を使用してデータベースから証明書を削除するには、以下を実行します。

1. インスタンスの証明書データベースディレクトリーを開きます。

```
/var/lib/pki/instance_name/alias
```

2. **-L** オプションを使用して **certutil** を実行し、データベースの証明書を一覧表示します。以下に例を示します。

```
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. **-D** オプションを指定して **certutil** を実行し、証明書を削除します。

```
certutil -D -d . -n certificate_nickname
```

以下に例を示します。

```
certutil -D -d . -n "ServerCert cert-instance_name"
```

4. 証明書を再度一覧表示し、証明書が削除されたことを確認します。

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

14.5. CA 証明書の信頼設定の変更

Certificate System サブシステムは、証明書データベース内の CA 証明書を使用して、TLS 対応の通信中に受信した証明書を検証します。

証明書データベースに保存されている CA の信頼設定を、一時的または永続的に変更する必要がある場合があります。たとえば、アクセスまたは侵害された証明書に問題がある場合、CA 証明書を信頼できないものとしてマークすると、その CA によって署名された証明書を持つエンティティーが Certificate System に対して認証されなくなります。問題が解決されると、CA は再び信頼できるとマークできます。

CA の信頼を永続的に解除するには、信頼データベースからその証明書を削除することを検討してください。手順は、「データベースからの証明書の削除」を参照してください。

14.5.1. コンソールからの信頼設定の変更

CA 証明書の信頼設定を変更するには、以下を実行します。

1. サブシステムコンソールを開きます。

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

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. **CA certificates** タブを選択します。
4. 変更する CA 証明書を選択し、**Edit** をクリックします。
5. **The Certificate chain is (un)trusted, are you sure you want to (un)trust it?** というプロンプトが開きます。

   yes をクリックすると、証明書チェーンの信頼設定が変更されます。**no** を押すと、元の信頼関係が保持されます。

14.5.2. certutil を使用した信頼設定の変更

certutil を使用して証明書の信頼設定を変更するには、以下を実行します。

1. インスタンスの証明書データベースディレクトリーを開きます。

   ```
   cd /var/lib/pki/instance_name/alias
   ```

2. **-L** オプションを使用して **certutil** を実行し、データベースの証明書を一覧表示します。以下に例を示します。

   ```
   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. **-M** オプションを使用して **certutil** を実行し、証明書の信頼設定を変更します。

   ```
   certutil -M -n cert_nickname -t trust -d .
   ```

   以下に例を示します。

   ```
   certutil -M -n "Certificate Authority - Example Domain" -t TCu,TCu,TCu -d .
   ```

4. 証明書を再度一覧表示し、証明書が削除されたことを確認します。

   ```
   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
   ```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

14.6. サブシステムによって使用されるトークンの管理

注記

TMS 上の本セクションにある機能は、評価でテストされていません。本セクションは参照用途としてのみ提供されています。

Certificate System は、トークンの 2 つのグループを管理します。PKI タスクを実行するためにサブシステムによって使用されるトークンと、サブシステムを通じて発行されるトークンです。これらの管理タスクは、特にサブシステムによって使用されるトークンを参照します。

14.6.1. トークンの検出

インストールまたは設定する Certificate System によってトークンを検出できるかどうかを確認するには、**TokenInfo** ユーティリティーを使用します。

```
TokenInfo /var/lib/pki/instance_name/alias
Database Path: /var/lib/pki/instance_name/alias
Found external module 'NSS Internal PKCS #11 Module'
```

このユーティリティーは、Certificate System にインストールされたトークンだけでなく、Certificate System で検出できるすべてのトークンを返します。

14.6.2. トークンの表示

Certificate System インスタンスに現在インストールされているトークンの一覧を表示するには、**modutil** ユーティリティーを使用します。

1. インスタンスの **alias** ディレクトリーを開きます。以下に例を示します。

```
cd /var/lib/pki/instance_name/alias
```

2. インストールされている PKCS #11 モジュールに関する情報と、**modutil** ツールを使用して、対応するトークンに関する情報を表示します。

```
modutil -dbdir . -nocertdb -list
```

14.6.3. トークンのパスワードの変更

サブシステムのキーペアと証明書を格納する内部または外部のトークンは、パスワードによって保護 (暗号化) されます。キーペアを復号する、またはキーペアにアクセスするには、トークンのパスワードを入力します。このパスワードは、トークンが最初にアクセスされたとき、通常は Certificate System のインストール時に設定されます。

サーバーのキーと証明書を保護するパスワードを定期的に変更することをお勧めします。パスワードを変更すると、誰かがパスワードを見つけるリスクを最小限に抑えることができます。トークンのパスワードを変更するには、**certutil** コマンドラインユーティリティーを使用します。

certutil の詳細は、http://www.mozilla.org/projects/security/pki/nss/tools/ を参照してください。

シングルサインオンパスワードキャッシュは、**password.conf** ファイルにトークンパスワードを保存します。このファイルは、トークンパスワードが変更されるたびに手動で更新する必要があります。

第15章 RED HAT ENTERPRISE LINUX 7.6 での時刻と日付の設定

このセクションには、Red Hat Enterprise Linux 7.6 で時刻と日付を設定する方法が含まれています。

システム時間は常に **協定世界時** (UTC) で維持され、必要に応じてアプリケーション内でローカル時間に変換されます。**ローカルタイム** は、**夏時間** (DST) を考慮に入れた現行タイムゾーンの実際の時刻です。

timedatectl ユーティリティーは、**systemd** システムおよびサービスマネージャーの一部として配布されており、システムクロック設定を確認および変更できます。

システムの現在時刻の変更

```
timedatectl set-time HH:MM:SS
```

HH は時間、*MM* は分、*SS* は秒 (すべて 2 桁) の数字に置き換えます。

現在日の変更

```
timedatectl set-time YYYY-MM-DD
```

YYYY は 4 桁の年に、*MM* と *DD* は 2 桁の月と日に置き換えます。

この時間の変更はオペレーティングシステムによって監査されます。詳細は、13.2.1.3 を参照してください。Red Hat Certificate System の計画、インストール、およびデプロイメントのガイドの時間変更イベントの監査

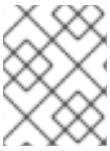
第16章 証明書システムの製品バージョンの特定

Red Hat Certificate System の製品バージョンは、**/usr/share/pki/CS_SERVER_VERSION** ファイルに保存されます。バージョンを表示するには、以下を実行します。

```
# cat /usr/share/pki/CS_SERVER_VERSION
Red Hat Certificate System 9.4 (Batch Update 3)
```

実行中のサーバーの製品バージョンを検索するには、ブラウザーから以下の URL にアクセスします。

- **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**



注記

各コンポーネントは個別のパッケージであるため、バージョン番号は異なります。上記は、現在実行中の各コンポーネントのバージョン番号を示しています。

第17章 RED HAT CERTIFICATE SYSTEM の更新

オペレーティングシステムが実行している Certificate System を更新するには、**yum update** コマンドを使用します。これにより、Certificate System およびオペレーティングシステムパッケージの更新がダウンロード、検証、およびインストールされます。Certificate System の更新および更新が成功したことの検証に関する詳細は、『Red Hat Certificate System Planning、Installation、and Deployment Guide (Common Criteria Edition)』 の 『Updating Certificate System Packages』 セクションを参照してください。

第18章 トラブルシューティング

この章では、Certificate System のインストール時に発生する一般的な使用上の問題のいくつかを説明します。

問： init スクリプトは OK ステータスを返しましたが、その CA インスタンスは応答しません。理由

答： これは起こらないはずです。通常 (常にではありませんが)、これは CA のリスナーの問題を示しますが、さまざまな原因が考えられます。**catalina.out**、**system**、**およびデバッグ** ログファイルでインスタンスに対して、発生したエラーを確認します。これは、いくつかの共通エラーを示しています。

1 つの状況は、CA の PID があり、プロセスが実行されているが、サーバーのリスナーが開かれていないことを示している場合です。これにより、Java 呼び出しクラスエラーが **catalina.out** ファイルに返されます。

```
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
```

これは、JSS または NSS の誤ったバージョンがあることを意味します。プロセスに **libnss3.so** が必要です。以下のコマンドでこれを確認します。

```
ldd /usr/lib64/libjss4.so
```

libnss3.so が見つからない場合は、**LD_LIBRARY_PATH** 変数の設定を解除し、CA を再起動します。

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

問： pkiconsole を開くことができません。標準出力 (stdout) で Java の例外が見られます。

答： これはおそらく、間違った JRE がインストールされているか、間違った JRE がデフォルトとして設定されていることを意味します。**alternatives --config java** を実行して、選択した JRE を確認します。Red Hat Certificate System には OpenJDK 1.8 が必要です。

問： pkiconsole の実行を試みましたが、標準出力 (stdout) でソケット例外が取得されました。理由

答： これは、ポートに問題があることを意味し、通常は、管理インターフェイスにアクセスするために間違ったポートが指定されたことを意味します。

ポートエラーは以下のようになります。

```
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)
```

問： 証明書を登録しようとしたら request is not submitted...Subject Name Not Found というエラーが表示される

答： これは、カスタム LDAP ディレクトリー認証プロファイルで最も頻繁に発生し、ディレクトリー操作が失敗したことを示しています。特に、作業 DN を作成できないために失敗しました。このエラーは CA の **debug** ログに表示されます。たとえば、このプロファイルは、ディレクトリーを認識しないカスタム属性 (***MYATTRIBUTE***) を使用します。

```
[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
```

サブジェクト DN で使用されるカスタムコンポーネント (属性、オブジェクトクラス、および未登録の OID) は、障害を引き起こす可能性があります。ほとんどの場合、RHC 2253 で定義されている X.509 属性は、カスタム属性ではなくサブジェクト DN で使用する必要があります。

問： 登録した証明書が公開されないのはなぜですか。

答： これは通常 CA の設定が間違っていることを示しています。エラーを検索する主要な場所は **debug** ログで、設定が間違っている場所を示しています。たとえば、以下にはマッパーに関連する問題があります。

```
[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
```

CA コンソールの **Publishing** タブで公開設定を確認します。この例では、この問題はマッピングパラメーターにあり、**既存の** LDAP 接尾辞を指している必要があります。

```
ca.publish.mapper.instance.LdapUserCertMap.dnPattern=UID=$subj.UID,dc=publish
```

問： リモートホストから pkiconsole ユーティリティーを開くにはどうすればいいですか

答： 特定の状況では、管理者が、リモートホストからの Certificate System サーバーに **pkiconsole** を開く場合があります。このため、管理者は Virtual Network Computing (VNC) 接続を使用できます。

1. Red Hat Certificate System サーバーなどで VNC サーバーを設定します。

重要

pkiconsole ユーティリティーは、Federal Information Processing Standard (FIPS) モードが有効になっているサーバーでは実行できません。Certificate System サーバーで FIPS モードが有効になっている場合は、Red Hat Enterprise Linux で別のホストを使用して VNC サーバーを実行します。

VNC サーバーのインストールの詳細は、『Red Hat システム管理者ガイド』の『VNC サーバー』セクションを参照してください。

2. VNC ビューアーを使用して、VNC サーバーを実行しているホストに接続します。詳細は、『Red Hat システム管理者ガイド』の『VNC ビューアー』セクションを参照してください。

3. VNC ウィンドウで **pkiconsole** ユーティリティーを開きます。以下に例を示します。

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

注記

VNC ビューアーは、さまざまなオペレーティングシステムで使用できます。ただし、Red Hat は、統合リポジトリーから Red Hat Enterprise Linux にインストールされた VNC ビューアーのみをサポートします。

問： LDAP サーバーが応答しないときにどうすればよいですか。

答： 内部データベースに使用されている RedHat Directory Server インスタンスが実行していない場合、接続の問題が発生した場合、または TLS 接続障害が発生した場合、それに依存するサブシステムインスタンスに接続できません。インスタンスのデバッグログは、特に LDAP 接続の問題を特定します。たとえば、LDAP サーバーがオンラインではない場合は、以下のようになります。

```
[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)
```

ケーブルが抜かれた、Red Hat Directory Server が停止した、重大なパケット損失が発生した、または TLS 接続を再作成できることを確認したなど、根本的なネットワークの問題を修正した後、問題の Certificate System インスタンスを停止して開始します。

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

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

第19章 サブシステムの制御およびメンテナーンス

この章では、Red Hat Certificate System サブシステムを制御 (開始、停止、再起動、およびステータスチェック) する方法と、一般的なメンテナーンス (ヘルスチェック) の推奨事項を説明します。

19.1. 起動、停止、再起動、およびステータスの取得

Red Hat Certificate System サブシステムインスタンスは、Red Hat Enterprise Linux 7 の **systemctl** ユーティリティーを使用して停止および起動できます。

インスタンスを起動するには、以下のコマンドを実行します。

```
# systemctl start unit_file@instance_name.service
```

インスタンスを停止するには、以下のコマンドを実行します。

```
# systemctl stop unit_file@instance_name.service
```

インスタンスを再起動するには、以下のコマンドを実行します。

```
# systemctl restart unit_file@instance_name.service
```

インスタンスのステータスを表示するには、以下のコマンドを実行します。

```
# systemctl status unit_file@instance_name.service
```

unit_file には、以下のいずれかの値を使用できます。

- **pki-tomcat**: ウォッチドッグが無効になっている場合
- **pki-tomcat-nuxwdog**: ウォッチドッグが有効な場合

19.2. サブシステムのヘルスチェック

管理者は、次のような考えられる障害を定期的に監視することが重要です。

- 完全なディスクが起因する監査障害
- HSM 接続の問題が原因の署名に失敗する
- LDAP サーバー接続の問題
- などになります。

セルフテストは、10章 *セルフテスト*で説明されているように要求で実行することもできます。

パート V. 参考資料

付録A 証明書プロファイルの入力および出力の参照

プロファイルの入力と出力は、証明書要求で予想される入力パラメーターと登録結果の出力形式を定義します。Red Hat Certificate System の他の多くのコンポーネントと同様に、プロファイルの入力と出力は、カスタマイズと柔軟性を提供するために JAVA プラグインとして実装されています。この付録では、デフォルトの入力プラグインおよび出力プラグインのリファレンスを提供します。

A.1. 入力の参照

入力により、特定の証明書プロファイルに関連付けられた登録ページに特定のフィールドが配置されます。証明書プロファイルに設定された入力は、適切なフィールドを使用して動的に登録ページを生成するために使用されます。これらの入力フィールドは、プロファイルが最終的な証明書を生成するために必要な情報を収集します。

A.1.1. CMC 証明書要求入力

CMC Certificate Request 入力は、Certificate Message over CMS (CMC) 証明書要求を使用した登録に使用され、要求フォームで送信されます。要求タイプは PKCS #10 または CRMF のいずれかである必要があり、唯一のフィールドは要求を貼り付けるための **Certificate Request** テキスト領域です。

例A.1

```
caCMCUserCert.cfg:input.i1.class_id=cmcCertReqInputImpl
```

A.1.2. nsHKeyCertRequest (Token Key) Input

トークン管理システム (TMS) では、トークンキー入力を使用して、エージェントが後で証明書ベースの認証に使用するハードウェアトークンのキーを登録します。

この入力により、以下のフィールドが登録フォームに置かれます。

- **トークンキー CUID**。このフィールドは、トークンデバイスの CUID (通常は一意のユーザー ID) を入力します。
- **Token Key User Public Key**。このフィールドには、トークンユーザーの公開鍵が含まれている必要があります。

例A.2

```
caTempTokenDeviceKeyEnrollment.cfg:input.i1.class_id=nsHKeyCertReqInputImpl
```

A.1.3. nsNKeyCertRequest (トークンユーザーキー) 入力

TMS では、Token User Key 入力は、ハードウェアトークンのユーザーのキーを登録するために使用され、エージェントは後で証明書ベースの認証にトークンを使用します。この入力により、以下のフィールドが登録フォームに置かれます。

- **トークンキーユーザー UID**。このフィールドは、トークンデバイスのユーザーの LDAP エントリーの UID を指定します。
- **Token Key User Public Key**。このフィールドには、トークンユーザーの公開鍵が含まれている必要があります。

例A.3

```
caTempTokenUserEncryptionKeyEnrollment.cfg:input.i1.class_id=nsNKeyCertReqInputImpl
```

A.1.4. 発行先の DN 入力

TMS では、Subject DN 入力を使用すると、特定の DN を入力して証明書のサブジェクト名として設定でき、入力によって単一の **Subject Name** フィールドが登録フォームに挿入されます。

例A.4

```
caAdminCert.cfg:input.i3.class_id=subjectDNInputImpl
```

A.1.5. Subject Alternative Name Extension 入力

TMS では、Subject Alternative Name Extension 入力は、Subject Alternative Name Extension Default プラグインとともに使用されます。これにより、管理者は、入力へのパターン **req_san_pattern_#** を使用して URI の番号付きパラメーター、**SubjectAltNameExt** 拡張を有効にできます。たとえば、以下が含まれる URI などです。

```
...&req_san_pattern_0=host0.Example.com&req_san_pattern_1=host1.Example.com
```

host0.Example.com および **host1.Example.com** を以下のプロファイルから **SubjectAltNameExt** 拡張機能に挿入します。

例A.5

```
input.i3.class_id=
input.i3.name=subjectAltNameExtInputImplsubjectAltNameExtInputImpl
…
policyset.serverCertSet.9.constraint.class_id=noConstraintImpl
policyset.serverCertSet.9.constraint.name=No Constraint
policyset.serverCertSet.9.default.class_id=subjectAltNameExtDefaultImpl
policyset.serverCertSet.9.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_0=$request.req_san_pattern_0$
policyset.serverCertSet.9.default.params.subjAltExtType_0=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.req_san_pattern_1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=$request.req_san_pattern_2$
```

```
policyset.serverCertSet.9.default.params.subjAltExtType_2=DNSName
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=2
```

A.2. 出力の参照

出力は、登録に成功したエンドユーザーへの応答です。

A.2.1. CMC 証明書出力

プロファイルフレームワークでは、出力プラグインを指定できます。CMC 要求を使用した証明書の登録の場合、出力は RFC 5272 に準拠した CMC 出力に暗黙的に設定されます。したがって、プロファイルで指定された出力値は無視されます。

A.2.2. nsNSKeyOutput

TMS では、このクラスは、トークンキーの DER エンコード証明書を返す出力プラグインを実装します。

例A.6 caTokenUserDelegateAuthKeyEnrollment.cfg

```
output.list=o1
output.o1.class=nsNKeyOutputImpl
```

付録B 証明書および CRL のデフォルト、制約、および拡張

この付録では、X.509 v3 で定義された標準の証明書拡張機能と、X.509 v3 が完成する前にリリースされた製品のバージョンで使用された Netscape で定義された拡張機能の両方を説明します。PKIX Part 1 の推奨事項など、特定の種類の証明書で使用する拡張機能の推奨事項を提供します。

重要

この付録は、Red Hat Certificate System で使用または設定可能なデフォルト、制約、証明書および CRL 拡張機能のリファレンスです。証明書および CRL 拡張の詳細な参照および説明は、RFC 5280 を参照してください。

この付録には以下のセクションが含まれます。

- 「デフォルトの参照」
- 「制約の参照」
- 「標準仕様の X.509 v3 証明書拡張機能リファレンス」
- 「CRL 拡張機能」

B.1. デフォルトの参照

デフォルトでは、証明書の内容の定義に使用されます。このセクションでは、事前定義されたデフォルト値を一覧表示し、定義します。

B.1.1. Authority Info Access 拡張のデフォルト

デフォルトでは、Authority Info Access 拡張をアタッチします。この拡張機能は、証明書を検証するアプリケーションが、証明書を発行した CA に関するオンライン検証サービスや CA ポリシーデータなどの情報にアクセスする方法を指定します。この拡張機能は、CA によって維持されている CRL の場所を直接指すために使用しないでください。CRL Distribution Points 拡張 「CRL Distribution Points 拡張機能のデフォルト」 は、CRL の場所への参照を提供します。

この拡張機能に関する一般的な情報は、「authorityInfoAccess」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.1 Authority Info Access Extension のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
Method_**n**	拡張機能が表示されている証明書を発行した CA に関する追加情報を取得するアクセス方法を指定します。以下の値のいずれかになります。 ● ocsp (1.3.6.1.5.5.7.48.1). ● calssuers (1.3.6.1.5.5.7.48.2) ● renewal (2.16.840.1.113730.16.1)
LocationType_**n**	証明書を発行した CA に関する追加情報を含む場所の一般名タイプを指定します。これは以下のいずれかのタイプになります。 ● DirectoryName ● DNSName ● EDIPartyName ● IPAddress ● OID ● RFC822Name ● URIName

パラメーター	説明
Location_**n**	証明書を発行した CA に関する追加情報を取得するためのアドレスまたは場所を指定します。 ● **directoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **dNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 ● **OID** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。 ● **URIName** の場合、値は、URL 構文およびエンコード規則に従った非相対ユニバーサルリソース識別子 (URI) である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://ocspResponder.example.com:8000**Certificate System は、IPv4 と IPv6 の IP アドレスの両方を許可します。
Enable_**n**	この場所を有効にするかどうかを指定します。**true** を選択してセットとしてマークします。**false** を選択して無効にします。

B.1.2. Authority Key Identifier 拡張機能のデフォルト

デフォルトでは、認証局キー識別子の拡張子が証明書に接続されます。拡張機能は、CA が証明書に署名するために使用する秘密鍵に対応する公開鍵を識別します。このデフォルトにはパラメーターがありません。この拡張を使用すると、公開鍵情報とともに証明書に含まれます。

このデフォルトには、次の制約があります。

- No Constraints は、「No Constraint」 を参照してください。

この拡張機能に関する一般的な情報は、「authorityKeyIdentifier」 を参照してください。

B.1.3. 認証トークンサブジェクト名のデフォルト

このプロファイルのデフォルトでは、認証トークン (AuthToken) オブジェクトの属性値に基づいてサブジェクト名が入力されます。

このデフォルトのプラグインは、ディレクトリーベースの認証マネージャーである **SharedToken** と連携します。

さらに、ディレクトリーベースの認証マネージャーは、発行する証明書のサブジェクト名を作成します。これは、**AuthToken** からのユーザーの DN 値を使用してサブジェクト名を形成します。

このデフォルトは、**AuthToken** からサブジェクト名を読み取り、それを証明書要求に配置して、最終的な証明書にサブジェクト名が含まれるようにするロールを果たします。

次の制約は、このデフォルトで定義できます。

- No Constraints は、「No Constraint」 を参照してください。

B.1.4. CMC ユーザー署名サブジェクト名デフォルト

このプロファイルのデフォルトは、CMC 要求の署名者の **subjectDN** に基づいてサブジェクト名を設定します。このデフォルトでは、次の制約を使用する必要があります。

- **CMCUserSignedSubjectNameConstraint**: 「CMC ユーザー署名サブジェクト名の制約」 を参照してください。

B.1.5. 基本的な制約拡張機能のデフォルト

デフォルトでは、基本制約の拡張を証明書にアタッチします。拡張機能は、証明書マネージャーが CA であるかどうかを特定します。この拡張機能は、証明書チェーンの検証プロセス中に、CA 証明書を識別し、証明書チェーンパスの長さの制約を適用するためにも使用されます。

この拡張機能に関する一般的な情報は、「basicConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Basic Constraints 拡張機能制約は、「Basic Constraints 拡張機能制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.2 基本的な制約エクステンションのデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
IsCA	証明書サブジェクトが CA であるかどうかを指定します。**true** の場合、サーバーは **PathLen** パラメーターをチェックして、証明書に指定したパスの長さを設定します。**false** の場合、サーバーは証明書のサブジェクトを CA 以外として処理し、**PathLen** パラメーターに指定された値を無視します。
PathLen	パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、**maxPathLen** パラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraint 拡張で指定されたパスの長さよりも短くする必要があります。**0** は、従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。**n** は、ゼロよりも大きい整数でなければなりません。従属 CA 証明書の下で許可される従属 CA 証明書の最大数を指定します。 フィールドが空白の場合、パスの長さはデフォルトで、発行者の証明書の Basic Constraint 拡張機能で設定されたパスの長さによって決定されます。発行者のパスの長さが無制限の場合は、下位 CA 証明書のパスの長さも無制限になります。発行者のパス長がゼロより大きい整数の場合、下位 CA 証明書のパス長は、発行者のパス長より 1 小さい値に設定されます。たとえば、発行者のパス長が 4 の場合、下位 CA 証明書のパス長は 3 に設定されます。

B.1.6. CA 有効性のデフォルト

このデフォルトでは、CA 証明書の登録または更新プロファイルにオプションが追加され、CA の署名証明書の有効期限の制約がバイパスされます。これは、発行された CA 証明書の有効期限が、発行された C A 署名証明書の有効期限よりも遅い可能性があることを意味します。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.3 CA 有効性のデフォルトパラメーター

パラメーター	説明
bypassCAnotafterrange	要求側の CA が、発行側の CA の有効期間を超えて有効期間が延長された証明書を要求できるかどうかのデフォルト値を設定します。
range	この証明書の絶対有効期間を日数で指定します。
startTime	現在の時間に基づいて有効期間が始まるタイミングを設定します。

B.1.7. 証明書ポリシーの拡張機能のデフォルト

デフォルトでは、Certificate Policy Mappings の拡張を証明書テンプレートに割り当てます。この拡張機能は、証明書が発行されたポリシーと証明書を使用できる目的を示す 1 つ以上のポリシーを定義します。デフォルトでは、最大 5 つのポリシーが定義されますが、値を変更できます。

この拡張機能に関する一般的な情報は、を参照してください。 「certificatePoliciesExt」

表B.4 証明書ポリシー拡張のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
numCertPolicies	定義できるポリシーの数を指定します。デフォルトは **5** です。
enable	**true** を選択してポリシーを有効にします。**false** を選択してポリシーを無効にします。
policyId	ポリシーの OID 識別子を指定します。
cpsURI.enable	拡張機能には、発行者 Certificate Practice Statement への URI を含めることができます。**true** を選択して URI を有効にします。**false** を選択して URI を無効にします。
CPSURI.value	この値は、CA によって公開される Certification Practice Statement (CPS) へのポインターです。ポインターは URI の形式になります。
usernotice.enable	拡張機能には、発行者の Certificate Practice Statement への URI を含めることも、ユーザー通知などの発行者情報をテキスト形式で埋め込むこともできます。ユーザー通知を有効にするには **true** を選択します。ユーザー通知を無効にするには、**false** を選択します。

パラメーター	説明
usernotice.noticeReference.noticeNumbers	この任意のユーザー通知パラメーターは、他の場所に保存されているメッセージを指す一連の番号です。
usernotice.noticeReference.organization	このオプションのユーザー通知パラメーターは会社名を指定します。
usernotice.explicitText.value	この任意のユーザー通知パラメーターには、証明書内のメッセージが含まれます。

B.1.8. CRL Distribution Points 拡張機能のデフォルト

デフォルトでは、CRL Distribution Points の拡張を証明書に割り当てます。この拡張機能は、証明書を検証しているアプリケーションが CRL 情報を取得して、証明書の失効ステータスを検証できる場所を識別します。

この拡張機能に関する一般的な情報は、「CRLDistributionPoints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.5 CRL Distribution Points 拡張設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
Type_**n**	CRL ディストリビューションポイントのタイプを指定します。許容値は **DirectoryName**、**URIName**、または **RelativeToIssuer** です。型は、**Name** フィールドの値に対応する必要があります。

パラメーター	説明
Name_**n**	CRL 配布ポイントの名前を指定します。名前は次のいずれかの形式にすることができます。 ● RFC 2253 構文の X.500 ディレクトリー名。名前は、**cn=CA Central, ou=Research Dept, o=Example Corporation, c=US** のように、証明書のサブジェクト名に似ています。 ● URIName (**http://testCA.example.com:80** など)。 ● CRL 発行者に対して相対的な場所を指定する RDN。この場合、**Type** 属性の値は **RelativeToIssuer** である必要があります。
Reasons_**n**	配布ポイントで保持される CRL で想定される失効理由を指定します。次の定数のコンマ区切りリストを提供します。 ● unused ● keyCompromise ● cACompromise ● affiliationChanged ● superseded ● cessationOfOperation ● certificateHold
IssuerType_**n**	ディストリビューション中に保持される CRL を署名した発行者の命名タイプを指定します。発行者名は以下のいずれかの形式になります。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName
IssuerName_**n**	CRL に署名した CRL 発行者の名前を指定します。許容値は次のとおりです。

パラメーター	説明
	• **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 • **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** • **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** • **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 • **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 • **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 • **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 • **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample** このパラメーターの値は、**issuerName** フィールドの値に対応している必要があります。

パラメーター	説明

B.1.9. Extended Key Usage 拡張機能のデフォルト

デフォルトでは、Extended Key Usage の拡張を証明書に登録します。

この拡張機能に関する一般的な情報は、「extKeyUsage」 を参照してください。

この拡張機能は、Key Usage 拡張機能に示されている基本的な目的に加えて、認証された公開鍵を使用できる目的を識別します。たとえば、Extended Key Usage 拡張が署名キーを特定した場合、Extended Key Usage の拡張では、キーの使用法を OCSP 応答のみに署名したり、Java™ アプレットだけに署名するのに絞り込むことができます。

表B.6 Extended Key Usage 拡張機能の PKIX 使用定義

使用方法	OID
サーバー認証	1.3.6.1.5.5.7.3.1
クライアント認証	1.3.6.1.5.5.7.3.2
コード署名	1.3.6.1.5.5.7.3.3
E メール	1.3.6.1.5.5.7.3.4
IPsec エンドシステム	1.3.6.1.5.5.7.3.5
IPsec トンネル	1.3.6.1.5.5.7.3.6
IPsec ユーザー	1.3.6.1.5.5.7.3.7
タイムスタンプ	1.3.6.1.5.5.7.3.8

Windows 2000 は、暗号化されたファイルシステム (EFS) と呼ばれる機能を使用して、ハードディスク上のファイルを暗号化できます。以下の 2 つの OID を持つ Extended Key Usage が含まれる証明書を使用します。

1.3.6.1.4.1.311.10.3.4 (EFS 証明書)

1.3.6.1.4.1.311.10.3.4.1 (EFS リカバリー証明書)

EFS リカバリー回復証明書は、ユーザーが秘密鍵を紛失し、その鍵で暗号化されたデータを使用する必要がある場合に、復元エージェントによって使用されます。Certificate System は、これら 2 つの OID をサポートし、これらの OID を含む Extended Key Usage 拡張機能を含む証明書を発行できるようにします。

通常のユーザー証明書は、リカバリー OID ではなく、EFS OID のみで作成する必要があります。

次の制約は、このデフォルトで定義できます。

- 拡張鍵の使用に関する制約。「拡張された鍵使用拡張制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。

- No Constraints は、「No Constraint」 を参照してください。

表B.7 Extended Key Usage 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
OID	キー使用目的を識別する OID を指定します。許容値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID です。たとえば、2.16.840.1.113730.1.99 です。キーの使用目的に応じて、OID は PKIX (表B.6「Extended Key Usage 拡張機能の PKIX 使用定義」 にリストされている) またはカスタム OID で指定できます。カスタム OID は、会社で使用するために予約された ID の登録済みサブツリーである必要があります。 Certificate System の評価とテストにカスタム OID を使用することは可能ですが、実稼働環境では、OID の定義と ID のサブツリーの登録に関する ISO 規則に準拠しています。

B.1.10. Freshest CRL 拡張機能のデフォルト

デフォルトでは、Freshest CRL 拡張を証明書に割り当てます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.8 Freshest CRL 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
PointEnable_**n**	**true** を選択してこのポイントを有効にします。**false** を選択してこのポイント を無効にします。
PointType_**n**	**DirectoryName** または **URIName** のいずれかの発行ポイントのタイプを指定します。

パラメーター	説明
PointName_**n**	• **pointType** が **directoryName** に設定されている場合、この値は証明書のサブジェクト名と同様に X.500 名である必要があります。たとえば、**cn=CACentral,ou=Research Dept,o=Example Corporation,c=US** となります。 • **pointType** が **URIName** に設定されている場合、名前は URI であるホストを指定する絶対パス名である必要があります。例: **http://testCA.example.com/get/crls/here/**。
PointIssuerName_**n**	CRL に署名した発行者の名前を指定します。名前は以下のいずれかの形式になります。 • **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 • **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** • **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** • **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 • **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 • **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFF**

パラメーター	説明
	F:FFFF:255.255.255.0、および FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000 になります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、1.2.3.4.55.6.5.99 です。 ● **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample** name の値は、**PointType_** で指定した形式に準拠する必要があります。
PointType_**n**	CRL に署名した CRL 発行者の一般的な名前タイプを指定します。許容値は次のとおりです。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName このパラメーターの値は、**PointIssuerName** フィールドの値に対応している必要があります。

B.1.11. 一般的な拡張機能のデフォルト

この拡張により、ユーザーが決定したデータで汎用拡張を作成できます。デフォルトでは、汎用拡張が正しく設定されます。

表B.9 一般的な拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
genericExtOID	拡張 OID 識別子を指定します。
genericExtData	拡張に含まれるバイナリーデータ。

B.1.12. Inhibit Any-Policy 拡張機能のデフォルト

CA に発行される証明書には、inhibit any-policy 拡張を使用できます。禁止ポリシー拡張は、値が { 2 5 29 32 0 } の特別な anyPolicy OID が、他の証明書ポリシーに対する明示的な一致とは見なされていないことを示します。

表B.10 Inhibit Any-Policy 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	**このポリシーは Critical とマークする必要があります。** この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
SkipCerts	このパラメーターで any-policy ポリシーが許可されなくなる前に、パスに表示される追加証明書の数を示します。**1** の値は、any-policy は、この証明書のサブジェクトによって発行された証明書で処理できますが、パス内の追加の証明書では処理できないことを示します。

B.1.13. Issuer Alternative Name 拡張機能のデフォルト

このデフォルトは、Issuer Alternative Name 拡張を証明書に割り当てます。Issuer Alternative Name 拡張は、インターネットスタイルのアイデンティティーを証明書発行者に関連付けるために使用されます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.11 Issuer Alternative Name 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
issuerAltExtType	これにより、使用する名前拡張のタイプが設定されます。これは以下のいずれかになります。 • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • IPAddress • OIDName
issuerAltExtPattern	拡張に追加する要求属性値を指定します。属性の値は、サポートされる一般名のタイプに準拠する必要があります。許容値は、証明書要求に含まれる要求属性です。 サーバーがリクエストの属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Issuer Alternative Name 拡張を証明書に追加しません。リクエストから適切な属性を使用して issuerAlternativeName を形成することができない場合は、トークン式なしでリテラル文字列を使用できます。たとえば、**認証局** です。

B.1.14. Key Usage 拡張機能のデフォルト

デフォルトでは、Key Usage 拡張機能が証明書に接続されます。拡張機能は、データ署名、キー暗号化、データ暗号化など、証明書に含まれるキーを使用する目的を指定します。これにより、キーペアの使用が所定の目的に制限されます。

この拡張機能に関する一般的な情報は、「keyUsage」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Key Usage 制約については、「主な使用拡張機能の制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.12 Key Usage 拡張機能のデフォルト設定パラメーター

パラメーター	説明

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
digitalSignature	TLS クライアント証明書と S/MIME 署名証明書への署名を許可するかどうかを指定します。 **t rue** を選択して設定します。
nonRepudiation	S/MIME 署名証明書に使用するかどうかを指定します。 **t rue** を選択して設定します。 **警告** このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。
keyEncipherment	サブジェクトの公開鍵を使用して秘密鍵と秘密鍵のどちらを暗号化するかを指定します。これは、TLS サーバー証明書および S/MIME 暗号化証明書に設定されます。 **t rue** を選択して設定します。
dataEncipherment	サブジェクトの公開鍵を使用して、キー資料とは対照的に、拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
keyAgreement	サブジェクトの公開鍵がキー合意に使用されるたびに拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
keyCertsign	公開鍵を使用して他の証明書の署名を検証するかどうかを指定します。この設定は CA 証明書に使用されます。**true** を選択してオプションを設定します。
cRLSign	CRL に署名する CA 署名証明書の拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
encipherOnly	公開鍵が鍵共有の実行中にデータを暗号化するためだけのものである場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合、**keyAgreement** も設定する必要があります。 **t rue** を選択して設定します。

パラメーター	説明
decipherOnly	公開鍵が鍵共有の実行中にデータを暗号化するためだけのものである場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合、**keyAgreement** も設定する必要があります。**true** を選択して設定します。

B.1.15. Name Constraints 拡張機能のデフォルト

このデフォルトでは、Name Constraints 拡張を証明書に割り当てます。この拡張機能は CA 証明書で使用され、証明書チェーン内の後続の証明書のサブジェクト名またはサブジェクト代替名を配置するネームスペースを示します。

この拡張機能に関する一般的な情報は、「nameConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトでは、許可されたサブツリーと除外されたサブツリーの両方に最大 5 つの場所が定義され、場所ごとにパラメーターが設定されます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.13 Name Constraints 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
PermittedSubtrees**n**.min	許可されるサブツリーの最小数を指定します。 - **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 - **0** は、サブツリーの最小数がゼロであることを指定します。 - **n** は、ゼロより大きい整数である必要があります。サブツリーの最小数を設定します。

パラメーター	説明
PermittedSubtreesmax_**n**	許可されるサブツリーの最大数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最大数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。許可されるサブツリーの最大数を設定します。
PermittedSubtreeNameChoice_**n**	拡張に含める許可されるサブツリーの一般な名前タイプを指定します。許容値は次のとおりです。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName
PermittedSubtreeNameValue_**n**	拡張に含める許可されるサブツリーの汎用名を指定します。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 ● **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める

パラメーター	説明
	必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 ● **IPAddress** の場合、Classless Inter-Domain Routing (CIDR) 表記に準拠する有効な IP アドレスを指定する必要があります。IPv4 アドレスは、**n.n.n.n** 形式、またはネットマスクを使用した **n.n.n.n/m** である必要があります (**10.34.3.133** または **110.34.3.133/24** など)。IPv6 アドレスも CIDR 表記に準拠する必要があります。ネットマスクには、**2620:52:0:2203:527b:9dff:fe56:4495/64** または **2001:db8::/64** があります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの **Realm|NameType|NameStrings** 形式になります。 **OtherName** の形式は **(type)oid,string** にする必要があります。例: **(IA5String)1.2.3.4,MyExample**
PermittedSubtreeEnable_**n**	**true** を選択して、このサブツリーエントリーを許可します。
ExcludedSubtrees**n**.min	除外されたサブツリーの最小数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最小数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。これにより、必要なサブツリーの最小数が設定されます。

パラメーター	説明
ExcludedSubtreeMax_**n**	除外されたサブツリーの最大数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最大数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。これにより、許可されるサブツリーの最大数が設定されます。
ExcludedSubtreeNameChoice_**n**	拡張に追加する除外されたサブツリーの一般名を指定します。許容値は次のとおりです。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName

パラメーター	説明
ExcludedSubtreeNameValue_**n**	拡張に含める許可されるサブツリーの汎用名を指定します。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 ● **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 ● **IPAddress** の場合、Classless Inter-Domain Routing (CIDR) 表記に準拠する有効な IP アドレスを指定する必要があります。IPv4 アドレスは、**n.n.n.n** 形式、またはネットマスクを使用した **n.n.n.n/m** である必要があります (**10.34.3.133** または **110.34.3.133/24** など)。IPv6 アドレスも CIDR 表記に準拠する必要があります。ネットマスクには、**2620:52:0:2203:527b:9dff:fe56:4495 /64** または **2001:db8::/64** があります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **OtherName** の場合、値は他の形式の名前です。これは、**PrintableString**、**IA5String**、**UTF8 String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample**

パラメーター	説明
ExcludedSubtreeEnable_n	**true** を選択して、この除外されたサブツリーエントリーを有効にします。

B.1.16. Netscape Certificate Type 拡張機能のデフォルト

警告

この拡張機能は廃止されています。代わりに、Key Usage または Extended Key Usage による証明書拡張を使用してください。

デフォルトでは、Netscape Certificate Type 拡張を証明書に割り当てます。拡張機能は、CA 証明書、サーバー TLS 証明書、クライアント TLS 証明書、S/MIME 証明書などの証明書タイプを識別します。これにより、証明書の使用が事前に決定された目的に制限されます。

B.1.17. Netscape Comment 拡張機能のデフォルト

警告

この拡張機能は廃止されています。

デフォルトでは、Netscape Comment 拡張機能が証明書にアタッチされます。拡張機能を使用して、証明書にテキスト形式のコメントを含めることができます。コメントを解釈できるアプリケーションは、証明書が使用または表示されるときにコメントを表示します。

この拡張機能に関する一般的な情報は、「netscape-comment」を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.14 Netscape Comment 拡張機能の設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
CommentContent	証明書に表示するコメントの内容を指定します。

B.1.18. デフォルト拡張機能なし

デフォルトを使用しない場合は、このデフォルトを使用して制約を設定できます。このデフォルトには設定がなく、デフォルトも設定されていませんが、使用可能なすべての制約を設定できます。

B.1.19. OCSP No Check 機能拡張のデフォルト

デフォルトでは、OCSP No Check 拡張機能が証明書に割り当てられます。拡張機能は、OCSP レスポンダー証明書でのみ使用する必要があり、OCSP 準拠のアプリケーションが、承認された OCSP レスポンダーが OCSP 応答に署名するために使用する証明書の失効ステータスを確認する方法を示します。

この拡張機能に関する一般的な情報は、「OCSPNocheck」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.15 OCSP No Check 機能拡張のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

B.1.20. Policy Constraints 拡張機能のデフォルト

このデフォルトでは、Policy Constraints 拡張を証明書に割り当てます。拡張機能は CA 証明書でのみ使用でき、パス検証を 2 つの方法で制限します。ポリシーマッピングを禁止するか、パス内の各証明書に受け入れ可能なポリシー識別子が含まれていることを要求します。デフォルトでは、**Req ExplicitPolicy** と **InhibitPolicy Mapping** の両方を指定できます。PKIX 標準では、証明書に存在する場合、拡張子が null シーケンスで設定されてはならないことが要求されています。少なくとも 2 つの指定されたフィールドが存在する必要があります。

この拡張機能に関する一般的な情報は、「policyConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.16 Policy Constraints 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
reqExplicitPolicy	明示的なポリシーが必要になる前に、パスで許可される証明書の総数を指定します。これは、受け入れ可能なポリシーが必要になる前に、下位 CA 証明書の下にチェーンできる CA 証明書の数です。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、明示的なポリシーが必要になる前に、パスで従属 CA 証明書を許可しないことを指定します。 ● **n** は、ゼロより大きい整数である必要があります。明示的なポリシーが必要になる前に、パスで許可される従属 CA 証明書の最大数を指定します。 この数は、証明書の検証中に使用される CA 証明書の数に影響します。このチェーンは、チェーンを検証して移動させるエンドエンティティー証明書で始まります。このパラメーターは、拡張がエンドエンティティーの証明書に設定されている場合は有効ではありません。
inhibitPolicyMapping	ポリシーマッピングが許可されなくなる前に、パスで許可される証明書の総数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、ポリシーマッピングが許可されなくなる前に、パスで従属 CA 証明書が許可されないことを指定します。 ● **n** は、ゼロより大きい整数である必要があります。ポリシーマッピングが許可されなくなる前に、パスで許可される従属 CA 証明書の最大数を指定します。たとえば、値を 1 にすると、ポリシーマッピングは、この証明書のサブジェクトによって発行された証明書で処理できますが、パス内の追加の証明書では処理できないことを示します。

B.1.21. Policy Mappers 拡張機能のデフォルト

このデフォルトでは、Policy Mappings の拡張を証明書にアタッチします。拡張機能は OID のペアを一覧表示し、それぞれのペアが 2 つの CA のポリシーステートメントを識別します。ペアリングは、ある CA の対応するポリシーが別の CA のポリシーと同等であることを示します。この拡張は、クロス証明書のコンテキストで役に立ちます。サポートされる場合、拡張は CA 証明書のみに含まれます。デフォルトでは、ポリシーステートメントに割り当てられた OID をペアにすることにより、ある CA のポリシーステートメントを別の CA のポリシーステートメントにマップします。

各ペアは、**issuerDomainPolicy** と **subjectDomainPolicy** の 2 つのパラメーターで定義されます。ペアは、発行した CA が、サブジェクト CA の **subjectDomainPolicy** と同等の **issuerDomainPolicy** を考慮することを意味します。CA の発行元のユーザーは、特定のアプリケーションの

issuerDomainPolicy を受け入れる可能性があります。ポリシーマッピングは、サブジェクト CA に関連付けられているどのポリシーが受け入れたポリシーと同等であるかをこれらのユーザーに通知します。

この拡張機能に関する一般的な情報は、「policyMappings」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.17 Policy Mappers 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
IssuerDomainPolicy_**n**	別の CA のポリシーステートメントとマッピングするために、発行元 CA のポリシーステートメントに割り当てられた OID を指定します。たとえば、1.2.3.4.5 です。
SubjectDomainPolicy_**n**	発行 CA のポリシーステートメントに対応するサブジェクト CA のポリシーステートメントに割り当てられた OID を指定します。たとえば、6.7.8.9.10 です。

B.1.22. Private Key Usage Period 拡張機能のデフォルト

Private Key Usage Period の拡張機能により、証明書発行者は、証明書自体に秘密鍵に異なる有効期間を指定できます。この拡張は、デジタル署名鍵の使用を目的としています。

表B.18 Private Key Usage Period の設定パラメーター

パラメーター	説明
Critical	この拡張は、常にクリティカルではないはずです。
puStartTime	このパラメーターは、開始時間を設定します。デフォルト値は **0** です。これは、拡張機能がアクティベートされた時点から有効期間を開始します。
puDurationDays	このパラメーターは、使用状況の期間を設定します。デフォルト値は **365** です。これは、拡張機能がアクティブ化されてから 365 日に有効期間を設定します。

B.1.23. 署名アルゴリズムのデフォルト

デフォルトでは、証明書要求に署名アルゴリズムがアタッチされます。このデフォルトは、証明書の署名に使用できる可能なアルゴリズムをエージェントに提示します。

次の制約は、このデフォルトで定義できます。

- アルゴリズム制約の署名は、「アルゴリズム制約の署名」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.19 署名アルゴリズムのデフォルト設定パラメーターの署名

パラメーター	説明
signingAlg	この証明書の作成に使用するデフォルトの署名アルゴリズムを指定します。**signingAlgsAllowed** パラメーターに含まれる値のいずれかを指定すると、エージェントはこの値をオーバーライドすることができます。
signingAlgsAllowed	この証明書の署名に使用できる署名アルゴリズムを指定します。アルゴリズムは以下のいずれか 1 つになります。 • SHA256withRSA • SHA384withRSA • SHA512withRSA • SHA256withEC • SHA384withEC • SHA512withEC

B.1.24. サブジェクト代替名の拡張機能のデフォルト

このデフォルトは、Subject Alternative Name 拡張を証明書に割り当てます。拡張機能は、電子メールアドレス、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、または URI などの追加の ID を証明書のサブジェクトにバインドします。標準では、証明書のサブジェクトフィールドに空のシーケンスが含まれている場合に、Subject Alternative 名の拡張子にサブジェクトの代替名が含まれている必要があり、拡張子にクリティカルのマークが付けられている必要があります。

ディレクトリーベースの認証方法の場合、Certificate System は任意の文字列およびバイト属性の値を取得し、それらを証明書要求に設定できます。これらの属性は、自動登録モジュールで定義された **ldapStringAttributes** および **ldapByteAttributes**フィールドに入力することで設定されます。

認証された属性 (LDAP データベースに格納されている属性を意味する) をこの拡張機能の一部にする必要がある場合は、**$request.X$** トークンの値を使用します。

サブジェクトの代替名にユニバーサル一意識別子 (UUID) を挿入するための追加の属性があります。このオプションは、バージョン 4 UUID の乱数を生成します。パターンは、追加 **subjAltExtSource** パラメーターで番号を生成するサーバーを参照することによって定義されます。

この例では、基本的なサブジェクトの代替名拡張のデフォルトが設定されています。

例B.1 サブジェクト代替名の拡張機能のデフォルト設定

```
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=$request.requester_email$
policyset.serverCertSet.9.default.params.subjAltExtType_0=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.SAN1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=http://www.server.example.com
policyset.serverCertSet.9.default.params.subjAltExtType_2=URIName
policyset.serverCertSet.9.default.params.subjAltExtType_3=OtherName
policyset.serverCertSet.9.default.params.subjAltExtPattern_3=(IA5String)1.2.3.4,$server.source$
policyset.serverCertSet.9.default.params.subjAltExtSource_3=UUID4
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_3=true
policyset.serverCertSet.9.default.params.subjAltExtType_4=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_4=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_4=
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=4
```

Subject Alternative Name 拡張機能のデフォルトは、プロファイル属性の証明書要求をチェックします。リクエストに属性が含まれる場合、プロファイルはその値を読み込み、拡張機能に設定します。LDAP ベースの認証が設定されている場合は、Subject Alternative Name 拡張機能のデフォルトで LDAP ディレクトリーから属性値を挿入することもできます。証明書に追加された拡張機能には、設定されたすべての属性が含まれています。

Subject Alternative Name 拡張機能のデフォルトで使用できる変数は、表B.20「サブジェクト代替名に値を挿入する変数」 に記載されています。

表B.20 サブジェクト代替名に値を挿入する変数

ポリシーセットトークン	説明
$request.auth_token.cn$	証明書を要求したユーザーの LDAP 共通名 (***cn***) 属性。
$request.auth_token.mail$	証明書を更新したユーザーの LDAP メール (***mail***) 属性の値。
$request.auth_token.tokenCertSubject$	証明書サブジェクト名。
$request.auth_token.uid$	証明書を要求したユーザーの LDAP ユーザー ID (***uid***) 属性。
$request.auth_token.user$	
$request.auth_token.userDN$	証明書を要求したユーザーのユーザー DN。

ポリシーセットトークン	説明
$request.auth_token.userid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.uid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.profileRemoteAddr$	要求するユーザーの IP アドレス。これは、クライアントに応じて IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。
$request.profileRemoteHost$	ユーザーのマシンのホスト名または IP アドレス。ホスト名は、**http://server.example.com** などの完全修飾ドメイン名およびプロトコルになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。
$request.requestor_email$	要求を送信したユーザーのメールアドレス。
$request.requestowner$	要求を送信した人。
$request.subject$	証明書が発行されるエンティティーのサブジェクト名 DN。たとえば、**uid=jsmith, e=jsmith@example.com** です。
$request.tokencuid$	登録の要求に使用されるスマートカードトークンのカード一意の ID (CUID)。
$request.upn$	Microsoft UPN。これには **(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.upn$** の形式があります。

ポリシーセットトークン	説明
$server.source$	サーバーに対し、サブジェクト名のバージョン 4 の UUID (乱数) コンポーネントを生成するように指示します。この値は常に **(IA5String)1.2.3.4,$server.source$** 形式になります。

1 つのエクステンションに複数の属性を設定できます。**subjAltNameNumGNs** パラメーターは、一覧表示された属性のうち、証明書に追加する必要があるものの数を制御します。このパラメーターはカスタムプロファイルに追加する必要があり、必要な数の属性を含めるためにデフォルトプロファイルで変更する必要がある場合があります。例B.1「サブジェクト代替名の拡張機能のデフォルト設定」 では、**subjAltNameNumGNs** が **3** に設定され、**RFC822Name**、**DNSName**、および **URIName** 名（一般名 **_0**、**_1**、および **_2**）を挿入します。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.21 サブジェクト代替名の拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
Pattern	拡張に追加する要求属性値を指定します。属性の値は、サポートされる一般名のタイプに準拠する必要があります。サーバーがリクエストの属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Subject Alternative Name 拡張を証明書に追加しません。許容値は、証明書要求に含まれる要求属性です。たとえば、**$request.requester_email$** です。
タイプ	request 属性の一般的な名前タイプを指定します。 • request-attribute の値が **local-part@domain** 形式のメールアドレスの場合は **RFC822Name** を選択します。たとえば、**jdoe@example.com** です。 • 証明書のサブジェクト名と同様に、request-attribute 値が X.500 ディレクトリー名の場合は **DirectoryName** を選択します。たとえば、**cn=Jane Doe, ou=Sales Dept, o=Example Corporation, c=US** です。 • request-attribute の値が DNS 名である場合に **DNSName** を選択します。たとえば、**corpDirectory.example.com** です。

パラメーター	説明
	• request-attribute の値が EDI party 名の場合は、**EDIPartyName** を選択します。例: Example Corporation。 • request-attribute 値が、**http** などの両方のスキームを含む非相対 URI である場合、およびホストの完全修飾ドメイン名または IP アドレスの場合は、**URIName** を選択します。例: **http://hr.example.com** です。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 • request-attribute 値が、ドットで区切られた数値コンポーネント表記で指定された有効な IP アドレスである場合は、**IPAddress** を選択します。たとえば、**128.21.39.40** です。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 • request-attribute 値が、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である場合は、**OIDName** を選択します。たとえば、**1.2.3.4.55.6.5.99** です。 • 他の形式の名前は **OtherName** を選択します。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample**
ソース	ID を生成するために使用する識別ソースまたはプロトコルを指定します。サポートされるソースは UUID4 で、UUID を作成する乱数を生成します。
コンポーネント数 (NumGN)	サブジェクトの別名に含める必要がある名前コンポーネントの数を指定します。

B.1.25. サブジェクトディレクトリー属性の拡張機能のデフォルト

デフォルトでは、Subject Directory 属性の拡張が証明書に割り当てます。Subject Directory Attributes 機能拡張は、証明書の件名に必要なディレクトリー属性の値をすべて伝えます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.22 サブジェクトディレクトリー属性の拡張機能のデフォルトの設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
名前	属性名。これは、***cn***、***mail*** などの LDAP ディレクトリー属性になります。
Pattern	拡張に追加する要求属性値を指定します。属性値は、属性の許可される値に準拠する必要があります。サーバーが属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Subject Directory Attributes 拡張を証明書に追加しません。たとえば、**$request.requester_email$** です。
Enable	その属性が証明書に追加できるかどうかを設定します。**true** を選択して属性を有効にします。

B.1.26. サブジェクト情報アクセス拡張機能のデフォルト

証明書テンプレートに Subject Information Access 拡張機能を設定する登録デフォルトポリシーを実装します。この拡張機能は、拡張機能が表示されている証明書のサブジェクトの情報とサービスにアクセスする方法を示します。

パラメーター	説明
Critical	この拡張はクリティカルではないはずです。
subjInfoAccessNumADs	証明書に含まれる情報アクセスセクションの数。
subjInfoAccessADMethod_n	アクセスメソッドの OID。

パラメーター	説明
subjInfoAccessADMethod_n	アクセスメソッドのタイプ。 • URIName • ディレクトリー名 • DNS 名 • EID パーティー名 • IP アドレス • OID 名 • RFC822Name
subjInfoAccessADLocation_n	タイプ subjInfoAccessADMethod_n を元にした場所 つまり、URI 名の URL。
subjInfoAccessADEnable_n	**true** を選択してこのエクステンションを有効にします。**false** を選択してこのエクステンションを無効にします。

B.1.27. Subject Key Identifier 拡張機能のデフォルト

デフォルトでは、サブジェクトキー識別子の拡張を証明書に割り当てます。この拡張機能は、特定の公開鍵を含む証明書を識別し、同じサブジェクト名を持つ複数の証明書の中から証明書を識別します。

この拡張機能に関する一般的な情報は、「subjectKeyIdentifier」 を参照してください。

有効にすると、拡張機能がまだ存在しない場合、プロファイルは登録要求に Subject Key Identifier Extensio 拡張機能を追加します。CRMF リクエストなど、リクエストに拡張機能が存在する場合は、デフォルトで拡張機能が置き換えられます。エージェントが手動登録要求を承認した後、プロファイルは、すでに存在する Subject Key Identifier Extension を受け入れます。

このデフォルトにはパラメーターがありません。この拡張を使用すると、公開鍵情報とともに証明書に含まれます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.28. サブジェクト名のデフォルト

デフォルトでは、サーバー側の設定可能なサブジェクト名を証明書要求に割り当てます。静的サブジェクト名は、証明書のサブジェクト名として使用されます。

次の制約は、このデフォルトで定義できます。

- Subject Name 制約の場合は、「Subject Name 制約」 を参照してください。

- Unique Subject Name 制約の場合は、「Unique Subject Name 制約」 を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.23 サブジェクト名のデフォルト設定パラメーター

パラメーター	説明
Name	この証明書のサブジェクト名を指定します。

UidPwdDirAuth プラグインから DNPATTERN 値を使用する証明書サブジェクト名を取得する必要がある場合は、Subject Name Default プラグインを使用するようにプロファイルを設定し、以下に示すように、***Name*** パラメーターを、**AuthToken**の SubjectName に置き換えます。

```
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.1.29. ユーザーキーのデフォルト

デフォルトでは、ユーザーが指定したキーを証明書要求に割り当てます。これは必須のデフォルトです。キーは登録要求の一部です。

次の制約は、このデフォルトで定義できます。

- キー制約については、「主要な制約」 を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.30. ユーザー署名アルゴリズムのデフォルト

このデフォルトは、証明書要求にユーザー指定の署名アルゴリズムを設定する登録デフォルトプロファイルを実装します。証明書プロファイルに含まれている場合、これにより、ユーザーは、制約セットに従って、証明書の署名アルゴリズムを選択できます。

署名アルゴリズムの選択肢を登録フォームに追加するための入力は提供されていませんが、この情報を含むリクエストを送信することは可能です。

次の制約は、このデフォルトで定義できます。

- アルゴリズム制約の署名は、「アルゴリズム制約の署名」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.31. ユーザーのサブジェクト名のデフォルト

デフォルトでは、ユーザーが指定したサブジェクト名を証明書要求に割り当てます。証明書プロファイルに含まれている場合、ユーザーは、設定された制約に従って、証明書のサブジェクト名を指定できます。この拡張機能は、証明書の発行時に元の証明書要求で指定されたサブジェクト名を保持します。

次の制約は、このデフォルトで定義できます。

- Subject Name 制約の場合は、「Subject Name 制約」 を参照してください。

- Unique Subject Name 制約の場合は、「Unique Subject Name 制約」を参照してください。
- No Constraints は、「No Constraint」を参照してください。

B.1.32. ユーザーの有効性のデフォルト

このデフォルトでは、ユーザーが指定した有効性が証明書要求に添付されます。証明書プロファイルに含まれている場合、ユーザーは設定された制約に従って有効期間を指定できます。このデフォルトプロファイルは、証明書が発行されたときに、元の証明書要求でそのユーザー定義の有効期間を保持します。

ユーザー指定の有効期限を登録フォームに追加するための入力は提供されていませんが、この情報を含むリクエストを送信することは可能です。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」を参照してください。

B.1.33. User Supplied Extension Default

User Supplied Extension Default クラスは、証明書要求でユーザーが定義した証明書拡張を証明書に入力します。プロファイルは証明書を登録する前に特定の拡張機能を必要とする可能性があるため、これには、ユーザーが特定の基準を満たす証明書要求を送信するか、特定の情報を提供する必要があります。

警告

この拡張機能のデフォルトの設定には、ユーザーが証明書要求で拡張機能を指定できるため、特に注意してください。このデフォルトを使用する場合、Red Hat は、拡張機能に対応する制約を使用して、User Supplied Extension Default の乱用を最小限に抑えることを強く推奨します。

ユーザー定義の拡張機能は、設定されている制約に対して検証されるため、拡張機能の種類を制限したり (Extension Constraint の制約を介して)、キーやその他の基本的な制約 (CA 証明書かどうかなど) のルールを設定したりできます。

CA は、次の 3 つの方法のいずれかで、User Supplied Extension Default を使用した登録を処理します。

- 拡張機能の OID が証明書要求とデフォルトの両方で指定されている場合、拡張機能は制約によって検証され、証明書に適用されます。
- 拡張機能の OID が要求で指定されているが、プロファイルの User Supplied Extension Default で指定されていない場合、ユーザー指定の拡張機能は無視され、証明書はその拡張機能なしで正常に登録されます。
- この拡張機能が対応する OID (拡張機能制約) を持つプロファイルに設定されている場合、そのプロファイルを介して処理される証明書要求には、指定された拡張機能を含める **必要があります**。そうでない場合、要求は拒否されます。

ユーザー定義の拡張を含む証明書 **要求** はプロファイルに送信する必要があります。ただし、証明書登録フォームには、ユーザーが提供する拡張機能を追加するための入力フィールドがありません。拡張機能を提供せずに証明書要求を送信すると失敗します。

例B.2「Extended Key Usage Extension の User Supplied Extension Default」 Extended Key Usage Constraint のあるプロファイルに User Supplied Extension Default を追加します。***userExtOID*** パラメーターで指定された OID は、Extended Key Usage Extension 用です。

例B.2 Extended Key Usage Extension の User Supplied Extension Default

```
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
```

例B.2「Extended Key Usage Extension の User Supplied Extension Default」 では、User Supplied Extension Default では、ユーザーは Extended Key Usage Extension (2.5.29.37)を指定できますが、制約により、ユーザー要求は TLS クライアント認証(1.3.6.1.5.5.7.3.2)と電子メール保護(1.3.6.1.5.5.7.3.4)の使用に制限されます。

プロファイルの編集については、Red Hat Certificate System Planning、Installation and Deployment Guide のファイルシステムで証明書プロファイルを直接作成および編集するセクションで説明されています。

例B.3 CSR の Multiple User Supplied Extension

RHCS 登録プロファイルフレームワークを使用すると、同じプロファイルで複数の User Supplied Extensions 拡張機能を定義できます。たとえば、以下の組み合わせを指定できます。

- Extended Key Usage Extension の場合:

```
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
```

- Key Usage Extension の場合:

  以下の形式を使用すると、拡張機能のパラメーターを適用するポリシーを適用できます。

  - CSR: **value = "true" になければなりません**
  - CSR: **value = "false" に存在してはなりません**
  - **オプション**: **value = "-"**

  以下に例を示します。

```
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
```

注記

ユーザー定義の拡張属性で CSR を作成する方法は、「**certutil** を使用したユーザー定義拡張による CSR の作成」 を参照してください。

B.1.34. 有効性のデフォルト

デフォルトでは、サーバー側の設定可能な有効期間を証明書要求に割り当てます。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.24 有効性のデフォルト設定パラメーター

パラメーター	説明
range	この証明書の有効期限を指定します。
startTime	現在の時間に基づいて有効期間が始まるタイミングを設定します。

B.2. 制約の参照

制約は、証明書の許容される内容とその内容に関連付けられた値を定義するために使用されます。このセクションでは、それぞれの完全定義を含む事前定義された制約を一覧表示します。

B.2.1. Basic Constraints 拡張機能制約

Basic Constraints 拡張制約は、証明書要求の基本制約がこの制約で設定された基準を満たしているかどうかを確認します。

表B.25 Basic Constraints 拡張機能の制約設定パラメーター

パラメーター	説明
basicConstraintsCritical	エクステンションは critical または noncritical のマークを付けるかどうかを指定します。この拡張機能をクリティカルとしてマークする場合は **true** を選択します。この拡張機能がクリティカルとしてマークされないようにするには、**false** を選択します。ハイフン **-** を選択しても、影響度が重大を意味します。
basicConstraintsIsCA	証明書サブジェクトが CA であるかどうかを指定します。**true** を選択すると、(CA である) このパラメーターに **true** の値を要求します。**false** を選択して、このパラメーターの **true** の値を無効にします。ハイフン **-** を選択すると、このパラメーターに制約を設定しないことを示します。
basicConstraintsMinPathLen	最小許容パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、このパラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraint 拡張で指定されたパスの長さよりも短くする必要があります。 **0** は、発行されている従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。 **n** は、ゼロよりも大きい整数でなければなりません。これは、使用されている従属 CA 証明書の下で許可される従属 CA 証明書の最小数です。

パラメーター	説明
basicConstraintsMaxPathLen	最大許容パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、このパラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraints 拡張で指定されたパスの長さよりも大きくする必要があります。 **0** は、発行されている従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。 **n** は、ゼロよりも大きい整数でなければなりません。これは、使用されている従属 CA 証明書の下で許可される従属 CA 証明書の最大数です。 フィールドが空白の場合、パスの長さはデフォルトで、発行者の証明書の Basic Constraint 拡張機能で設定されたパスの長さによって決定されます。発行者のパスの長さが無制限の場合は、下位 CA 証明書のパスの長さも無制限です。発行者のパス長がゼロより大きい整数の場合、下位 CA 証明書のパス長は、発行者のパス長より 1 小さい値に設定されます。たとえば、発行者のパス長が 4 の場合、下位 CA 証明書のパス長は 3 に設定されます。

B.2.2. CA Validity 制約

CA 有効性制約は、証明書テンプレートの有効期間が CA の有効期間内にあるかどうかをチェックします。証明書の有効期間が CA 証明書の有効期間外である場合は、制約が拒否されます。

B.2.3. 拡張された鍵使用拡張制約

Extended Key Usage 拡張制約は、証明書の Extended Key Usage 拡張機能がこの制約で設定された基準を満たしているかどうかを確認します。

重要

キー使用法拡張と拡張キー使用法拡張の一貫性を維持する必要があります。詳細については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Key Usage および Extended Key Usage Consistency』 セクションを参照してください。

表B.26 拡張された主な使用拡張制約設定パラメーター

パラメーター	説明
exKeyUsageCritical	**true** に設定すると、エクステンションは重要であるとマークできます。**false** に設定すると、拡張は重要でないとしてマークできます。
exKeyUsageOIDs	キーの使用目的を特定する許容可能な OID を指定します。複数の OID をコンマ区切りの一覧に追加できます。

B.2.4. 拡張制約

この制約は、一般的な拡張制約を実装します。拡張機能が存在するかどうかを確認します。

表B.27 拡張制約

パラメーター	説明
extCritical	エクステンションは critical または noncritical のマークを付けるかどうかを指定します。**true** を選択して拡張機能を重要 (Critical) とマークします。**false** を選択して、非クリティカルにマークします。**-** を選択して、優先なしを強制します。
extOID	制約を渡すために証明書に存在する必要がある拡張の OID。

B.2.5. 主要な制約

この制約は、RSA キーのキーのサイズと、EC キーの楕円曲線の名前を確認します。RSA キーと一緒に使用すると、***KeyParameters*** パラメーターには、有効なキーサイズのコンマ区切りリストが含まれ、EC キーの場合は ***KeyParameters*** パラメーターには、使用可能な ECC 曲線のコンマ区切りのリストが含まれています。

表B.28 キー制約の設定パラメーター

パラメーター	説明
keyType	キーの種類を指定します。これはデフォルトで **-** に設定されており、RSA キーシステムを使用します。rsa と ec を選択できます。キータイプが指定され、システムで識別されていない場合、制約は拒否されます。

パラメーター	説明
KeyParameters	特定のキーパラメーターを定義します。キーに設定されるパラメーターは、***keyType*** パラメーターの値によって異なります (つまり、キータイプによって異なります)。 ● RSA 鍵では、***KeyParameters*** パラメーターに有効な鍵サイズのコンマ区切りリストが含まれます。 ● ECC キーでは、***KeyParameters*** パラメーターに、利用可能な ECC 曲線のコンマ区切りリストが含まれます。

B.2.6. 主な使用拡張機能の制約

Key Usage 拡張制約は、証明書要求のキー使用制約がこの制約で設定された基準を満たしているかどうかを確認します。

重要

キー使用法拡張と拡張キー使用法拡張の一貫性を維持する必要があります。詳細については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Key Usage および Extended Key Usage Consistency』 セクションを参照してください。

表B.29 主な使用拡張制約設定パラメーター

パラメーター	説明
keyUsageCritical	**true** を選択して、この拡張機能を重要としてマークします。**false** を選択して、非クリティカルにマークします。設定しない場合は **-** を選択します。
keyUsageDigitalSignature	TLS クライアント証明書と S/MIME 署名証明書に署名するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

パラメーター	説明
kleyUsageNonRepudiation	S/MIME 署名証明書を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。 **警告** このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。
keyEncipherment	TLS サーバー証明書と S/MIME 暗号化証明書の拡張子を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageDataEncipherment	キーマテリアルの代わりに、サブジェクトの公開キーを使用してユーザーデータを暗号化するときに、拡張子を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageKeyAgreement	サブジェクトの公開鍵がキー合意に使用されるたびに拡張を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageCertsign	この機能がすべての CA 署名証明書に適用されるかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

パラメーター	説明
keyUsageCRLSign	CRL に署名するのに使用する CA 署名証明書の拡張を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageEncipherOnly	公開鍵を使用してデータの暗号化のみに使用する場合に、拡張機能を設定するかどうかを指定します。このビットが設定されている場合は、**keyUsageKeyAgreement** も設定する必要があります。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageDecipherOnly	公開鍵をデータの解読にのみ使用する場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合は、**keyUsageKeyAgreement** も設定する必要があります。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

B.2.7. Netscape Certificate Type 拡張機能の制約

警告

この制約は廃止されました。Netscape Certificate Type 拡張制約を使用する代わりに、Key Usage 拡張または Extended Key Usage 拡張を使用します。

Netscape Certificate Type 拡張制約は、証明書要求の Netscape Certificate Type 拡張がこの制約で設定された基準を満たしているかどうかをチェックします。

B.2.8. No Constraint

この制約は、制約が実装されていません。デフォルトとともに選択すると、そのデフォルトには制約は含まれません。

B.2.9. Renewal Grace Period 制約

Renewal Grace Period Constraint は、ユーザーが有効期限に基づいて証明書を更新できる時期に関するルールを設定します。たとえば、ユーザーは、有効期限が切れる前の特定の時間まで、または有効期限後の特定の時間を過ぎると、証明書を更新できません。

この制約を使用するときに覚えておくべき重要なことの 1 つは、この制約が更新プロファイルではなく、**元の登録プロファイル** に設定されていることです。更新猶予期間のルールは元の証明書の一部であり、引き継がれ、その後の更新に適用されます。

この制約は、No Default 拡張機能でのみ使用できます。

表B.30 Renewal Grace Period 制約設定パラメーター

パラメーター	説明
renewal.graceAfter	証明書の期限が切れた **後**、更新用に提出できる期間を日数単位で設定します。証明書の有効期限が切れると、更新要求は拒否されます。値を指定しない場合、制限はありません。
renewal.graceBefore	証明書の期限が切れる **前**、更新用に提出できる期間を日数単位で設定します。証明書が有効期限にそれほど近くない場合、更新要求は拒否されます。値を指定しない場合、制限はありません。

B.2.10. アルゴリズム制約の署名

署名アルゴリズム制約は、証明書要求の署名アルゴリズムがこの制約で設定された基準を満たしているかどうかを確認します。

表B.31 アルゴリズム制約設定パラメーターの署名

パラメーター	説明
signingAlgsAllowed	証明書の署名に指定できる署名アルゴリズムを設定します。アルゴリズムは以下のいずれか 1 つになります。 • SHA256withRSA • SHA384withRSA • SHA512withRSA • SHA256withEC • SHA384withEC • SHA512withEC

B.2.11. Subject Name 制約

Subject Name 制約は、証明書要求のサブジェクト名が基準を満たしているかどうかを確認します。

表B.32 Subject Name 制約設定パラメーター

パラメーター	説明
Pattern	サブジェクト DN を構築する正規表現または他の文字列を指定します。

Subject Name 名と正規表現

Subject Name Constraint の正規表現は、正規表現を照合するための Java 機能によって照合されます。これらの正規表現の形式は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html に記載されています。これにより、アスタリスク (*) などのワイルドカードは、任意の数の文字とピリオド (.) が、任意のタイプの文字を検索します。

たとえば、サブジェクト名制約のパターンが **uid=.*** に設定されている場合、証明書プロファイルフレームワークは、証明書要求のサブジェクト名がパターンと一致するかどうかを確認します。**uid=user, o=Example, c=US** のようなサブジェクト名は、**uid=.*** というパターンに対応します。サブジェクト名 **cn=user, o=example,c=US** はパターンを満たしません。**uid=.*** は、サブジェクト名は、***uid*** 属性で始まる必要があることを意味します。ピリオドアスタリスク (.*) ワイルドカードを使用すると、***uid*** の後に任意のタイプと文字数文字を続けることができます。

.*ou=Engineering.* などの内部パターンが必要になる場合があります。これには、その前後で任意の種類の文字列を持つ **ou=Engineering** 属性が必要になります。これは **cn=jdoe,ou=internal,ou=west coast,ou=engineering,o="Example Corp",st=NC** と同様に **uid=bjensen,ou=engineering,dc=example,dc=com** と一致します。

最後に、オプションの間にパイプ記号 (|) を設定することで、ある文字列または別の文字列のリクエストを許可することもできます。たとえば、**ou=engineering,ou=people** または **ou=engineering,o="Example Corp"** のいずれかが含まれるサブジェクト名を許可するには、パターンは **.*ou=engineering,ou=people.* | .*ou=engineering,o="Example Corp".*** のいずれかになります。

注記

ピリオド (.) などの特殊文字を使用するパターンを作成する場合は、バックスラッシュ (\) で文字をエスケープします。たとえば、文字列 **o="Example Inc."** を検索するには、パターンを **o="Example Inc\."** に設定します。

証明書の要求における Subject Name および UID または CN

サブジェクト DN を構築するために使用されるパターンは、証明書を要求する人の CN または UID に基づくこともできます。Subject Name Constraint は、証明書要求の DN で認識する CN (または UID) のパターンを設定し、Subject Name Default は、その CN に、事前定義されたディレクトリーツリーを使用して証明書のサブジェクト DN を作成します。

たとえば、証明書要求の CN を使用するには、次を実行します。

```
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. Unique Key 制約

この制約は、公開鍵が一意であることを確認します。

表B.33 Unique Key の制約パラメーター

パラメーター	説明
allowSameKeyRenewal	公開鍵は一意でなく、**かつ** サブジェクト DN が既存の証明書と一致し、このパラメーターが **true** に設定されている場合、リクエストは更新と見なされ、受け入れられます。ただし、公開鍵が重複していて、既存の Subject DN と一致しない場合、要求は拒否されます。 このパラメーターが **false** に設定されていると、重複した公開鍵リクエストは拒否されます。

B.2.13. Unique Subject Name 制約

Unique Subject Name 制約は、サーバーが同じサブジェクト名で複数の証明書を発行することを制限します。証明書要求が送信されると、サーバーは自動的にニックネームを他の発行された証明書のニックネームと照合します。この制約は、エンドエンティティーのページからの証明書の登録と更新に適用できます。

1 つの証明書の有効期限が切れているか取り消されていない限り (保留されていない場合)、証明書に同じサブジェクト名を付けることはできません。したがって、アクティブな証明書はサブジェクト名を共有できません。ただし、例外が 1 つあります。証明書のキー使用ビットが異なる場合は、使用法が異なるため、同じサブジェクト名を共有できます。

表B.34 Unique Subject Name 制約設定パラメーター

パラメーター	説明
enableKeyUsageExtensionChecking	キーの使用設定が異なる限り、証明書が同じサブジェクト名を持つことを許可するオプションの設定。これは **true** または **false** です。デフォルトは **true** で、重複したサブジェクト名を許可します。

B.2.14. CMC ユーザー署名サブジェクト名の制約

CMC ユーザーが署名したサブジェクト名の制約は、CMC 要求に署名したユーザーに発行される証明書を制限します。**CMCUserSignedSubjectNameDefault** で使用します。

B.2.15. Validity 制約

Validity 制約は、証明書要求の有効期間が基準を満たしているかどうかを確認します。

提供されるパラメーターは、適切な値でなければなりません。たとえば、すでに経過した時間を提供する ***notBefore*** パラメーターは受け入れず、 ***notBefore*** 時間よりも早い時間を提供する ***notAfter*** パラメーターは受け入れません。

表B.35 Validity 制約の設定パラメーター

パラメーター	説明
range	有効期間の範囲。日数を設定する整数です。***notBefore*** 時間と ***notAfter*** 時間の差 (日数) は範囲値よりも短くする必要があります。そうでない場合、この制約は拒否されます。
notBeforeCheck	範囲が猶予期間内ではないことを確認します。ブール値パラメーター ***NotBeforeCheck*** が true に設定されている場合、システムは、***notBefore***時間が、現在の時間に ***notBeforeGracePeriod*** 値を加えた値より大きくないことを確認します。***notBeforeTime*** が、現在の時間と ***notBeforeGracePeriod*** 値の間になければ、この制約は拒否されます。
notBeforeGracePeriod	***notBefore*** 時間後の猶予期間 (秒単位)。***notBeforeTime*** が、現在の時間と ***notBeforeGracePeriod*** 値の間になければ、この制約は拒否されます。この制約は、***notBeforeCheck*** パラメーターが true に設定されている場合にのみ確認されます。
notAfterCheck	指定された時間が期限切れの期間後にないかどうかを確認します。ブール値パラメーターが ***notAfterCheck*** を true に設定されている場合、システムは ***notAfter*** 時間は現在の時間より大きくありません。現在の時間がこの ***notAfter*** 時間を超えると、この制約は拒否されます。

B.3. 標準仕様の X.509 V3 証明書拡張機能リファレンス

X.509 v3 証明書には、証明書に任意の数の追加フィールドを追加できる拡張フィールドが含まれています。証明書拡張機能は、代替サブジェクト名や使用制限などの情報を証明書に追加する方法を提供します。Red Hat Directory Server や Red Hat Certificate System などの古い Netscape サーバーは、PKIX パート 1 標準が定義される前に開発されたため、Netscape 固有の拡張機能が必要です。

以下は、X.509 v3 拡張機能が含まれる証明書のセクションの例です。Certificate System は、以下に示すように、読み取り可能な pretty-print 形式で証明書を表示できます。この例のように、証明書の拡張子は順番に表示され、証明書ごとに特定の拡張子のインスタンスが 1 つだけ表示される場合があります。たとえば、証明書に含まれるサブジェクトキー識別子の拡張子は 1 つだけです。これらの拡張機能をサポートする証明書には、バージョン **0x2** があります (これは、バージョン 3 に対応します)。

例B.4 Pretty-Print 証明書拡張機能の例

```
Certificate:
 Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: PKCS #1 SHA-256 With RSA Encryption
    Issuer: "CN=CA Signing Certificate,OU=pki-tomcat,O=EXAMPLE"
    Validity:
```

```
        Not Before: Fri Feb 22 19:06:56 2019
        Not After : Tue Feb 22 19:06:56 2039
    Subject: "CN=CA Signing Certificate,OU=pki-tomcat,O=EXAMPLE"
    Subject Public Key Info:
        Public Key Algorithm: PKCS #1 RSA Encryption
        RSA Public Key:
            Modulus:
                dd:6d:ad:02:10:43:12:ad:ec:6c:10:82:b3:bc:ec:6d:
                4b:e9:46:bc:a3:19:63:15:86:cf:6d:62:43:92:6b:a6:
                3d:72:54:4b:4f:d5:ad:a9:1d:76:8d:1c:e9:15:24:10:
                a1:03:1e:1b:14:5e:08:0a:0f:5b:02:aa:e9:3f:85:e1:
                d4:a6:01:1e:58:ab:7b:f2:67:32:f4:95:3d:35:9c:76:
                3a:cb:3b:ef:e3:7d:32:04:bb:35:46:68:bd:21:0c:16:
                b6:63:aa:e7:bb:cd:0f:55:66:21:09:e6:a6:f7:4c:fd:
                af:c8:a6:d1:98:03:aa:89:b8:76:e7:dd:df:2b:23:c5:
                b3:06:16:1d:4a:13:8b:0b:56:0c:d5:a2:9a:22:5e:7d:
                08:af:e4:bf:a0:f6:28:ee:ae:0f:2c:b2:4d:2a:09:5b:
                6f:32:2e:05:3a:3b:92:5d:d6:1d:69:95:09:0d:f4:b8:
                52:ac:48:0f:a8:4f:0a:22:1b:01:4c:d2:79:89:e0:bc:
                cd:1c:84:f8:88:e6:92:16:ed:08:ad:6d:9c:17:8d:70:
                92:bd:18:74:1a:31:5f:9b:f7:eb:f7:6e:f8:9a:e6:37:
                fe:7a:c6:07:9b:8a:6c:e8:5b:77:7c:37:e0:66:39:72:
                62:5d:5d:d0:65:a2:d9:b0:7f:d3:ba:ed:4b:42:89:47
            Exponent: 65537 (0x10001)
    Signed Extensions:
        Name: Certificate Authority Key Identifier
        Key ID:
            88:fb:c7:45:a8:b8:e9:74:ab:71:a2:ab:ce:4e:26:b9:
            a5:97:dc:05

        Name: Certificate Basic Constraints
        Critical: True
        Data: Is a CA with no maximum path length.

        Name: Certificate Key Usage
        Critical: True
        Usages: Digital Signature
                Non-Repudiation
                Certificate Signing
                CRL Signing

        Name: Certificate Subject Key ID
        Data:
            88:fb:c7:45:a8:b8:e9:74:ab:71:a2:ab:ce:4e:26:b9:
            a5:97:dc:05

        Name: Authority Information Access
        Method: PKIX Online Certificate Status Protocol
        Location:
            URI: "http://localhost.localdomain:8080/ca/ocsp"

Signature Algorithm: PKCS #1 SHA-256 With RSA Encryption
Signature:
    6b:ed:d8:2b:de:40:a4:14:dd:e8:ce:52:2d:40:0a:f1:
    88:57:36:3b:7f:c4:e8:77:2b:95:e9:60:fd:57:9b:c2:
    2d:17:a2:67:4e:c0:23:00:7a:2c:ef:5f:12:13:05:cc:
```

```
        9e:d7:4f:70:55:68:88:eb:29:34:94:cd:59:a6:92:31:
        c6:36:74:dd:e5:a2:1f:b1:9e:6d:f0:41:95:c2:7f:4c:
        38:46:62:d9:f3:27:f4:a3:a7:f3:a2:ba:1c:e5:77:4a:
        d3:2d:50:10:47:03:2e:4f:f2:ef:75:92:36:d8:99:6d:
        f6:ef:f5:ee:17:70:c2:e0:c1:a1:26:fa:00:e2:ec:35:
        d5:11:4d:df:66:8d:3c:84:fa:72:ff:47:a5:95:08:c2:
        80:e6:19:60:ab:51:d6:f1:aa:ac:72:77:d0:01:97:1f:
        13:f0:c9:55:09:4d:d9:62:5b:bc:4a:21:5a:af:77:cb:
        4e:cf:48:aa:3d:fc:f6:5e:c8:e2:e0:e3:58:58:40:39:
        2b:9c:15:d3:65:62:d0:96:1b:35:3f:6e:35:96:ae:36:
        c2:6c:2b:46:e8:a3:d3:52:21:f0:47:5a:73:5e:1a:b0:
        99:2f:5d:1b:bc:a1:81:65:68:16:08:e8:3e:2f:5e:32:
        79:ca:8e:25:e5:78:a1:fc:cd:c0:b3:aa:83:02:18:43
    Fingerprint (SHA-256):

2B:2F:05:59:12:F7:A4:6D:DE:22:43:82:59:EC:9F:45:AD:6C:1E:0A:63:6B:79:57:B1:34:3E:1B:BA:D2
:13:AC
    Fingerprint (SHA1):
        E1:87:42:85:AF:07:6C:B2:5F:07:CB:50:4D:49:17:AB:43:99:31:F7

    Mozilla-CA-Policy: false (attribute missing)
    Certificate Trust Flags:
        SSL Flags:
            Valid CA
            Trusted CA
            Trusted Client CA
        Email Flags:
            Valid CA
            Trusted CA
        Object Signing Flags:
```

オブジェクト識別子 (OID) は、証明書の拡張子や会社の証明書運用明細書など、一意のオブジェクトを識別する数字の文字列です。Certificate System には、サーバーが発行する証明書に X.509 証明書拡張を追加できるようにする拡張固有のプロファイルプラグインモジュールのセットが付属しています。一部の拡張機能には、OID を指定するためのフィールドが含まれています。

PKIX 標準では、証明書で使用される拡張子やステートメントなどのすべてのオブジェクトを OID の形式で含めることを推奨しています。これにより、共有ネットワーク上の組織間の相互運用性が促進されます。共有ネットワークで使用される証明書が発行される場合は、OID 接頭辞を適切な登録機関に登録してください。

OID は、国際標準組織 (ISO) 登録機関によって制御されます。場合によっては、この権限は ISO から地域の登録機関に委任されます。米国では、American National Standards Institute (ANSI) がこの登録を管理します。

別の組織に登録されている OID を使用したり、OID の登録に失敗したりすると、状況によっては法的な結果が生じる可能性があります。登録には手数料がかかる場合があります。詳細は、適切な登録認証局にお問い合わせください。

カスタムオブジェクトの OID を定義または割り当てるには、会社の *アーク*、つまり民間企業の OID を知ってください。会社にアーカイブがない場合は、ファイルを取得する必要があります。http://www.alvestrand.no/objectid/ には、OID の登録および使用に関する詳細情報が記載されています。

たとえば、**Netscape Certificate Comment** という名前の拡張の Netscape 定義の OID は

2.16.840.1.113730.1.13 です。この拡張機能に割り当てられた OID は階層的であり、以前の Netscape 社のアーク **2.16.840.1** が含まれています。OID 定義エントリーは http://www.alvestrand.no/objectid/2.16.840.1.113730.1.13.html です。

OID 拡張が証明書に存在し、クリティカルとマークされている場合、証明書を検証するアプリケーションは、任意の修飾子を含めて拡張を解釈できる必要があります。そうでない場合は、証明書を拒否する必要があります。すべてのアプリケーションが OID の形式で埋め込まれた会社のカスタム拡張機能を解釈できる可能性は低いため、PKIX 標準では、拡張機能を常に非クリティカルではないとマークすることを推奨しています。

このセクションでは、インターネット X.509 バージョン 3 標準の一部として定義されている拡張タイプを要約し、PKIX ワーキンググループが推奨するタイプを紹介します。

この参照では、各証明書に関する重要な情報をまとめています。詳細については、ITU から入手できる X.509 v3 標準と、RFC 5280 で入手できる 『Internet X.509 Public Key Infrastructure - Certificate and CRL Profile (RFC 5280)』 の両方を参照してください。拡張機能の説明は、拡張機能について説明している標準ドラフトの RFC とセクション番号を参照しています。各拡張機能のオブジェクト識別子 (OID) も提供しています。

証明書の各拡張は、クリティカルまたは非クリティカルとして指定できます。Web ブラウザーなどの証明書を使用するシステムは、認識できない重要な拡張子に遭遇した場合、証明書を拒否する必要があります。ただし、重要でない拡張子は、認識されない場合は無視できます。

B.3.1. authorityInfoAccess

Authority Information Access 拡張機能は、証明書の発行者に関する情報にアクセスする方法と場所を示します。エクステンションには **accessMethod** および **accessLocation** フィールドが含まれます。**accessMethod** は、**accessLocation** という名前の発行者に関する情報のタイプおよび形式を OID に指定します。

PKIX Part 1 は、**accessMethod** (**id-ad-calssuers**) を 1 つ定義して、この拡張機能を使用して、証明書の発行者よりも CA チェーンの上位にある証明書を発行した CA のリストを取得します。**accessLocation** フィールドには、通常、リストの取得に使用される場所とプロトコル (LDAP、HTTP、または FTP) を示す URL が含まれます。

RFC 2560 で入手可能な Online Certificate Status Protocol (RFC 2560) は、OCSP を使用して証明書を検証するために *accessMethod* (**id-ad-ocsp**) を定義します。次に、**accessLocation** フィールドには、証明書を検証できる OCSP レスポンダーへのアクセスに使用される場所とプロトコルを示す URL が含まれます。

OID

1.3.6.1.5.5.7.1.1

Criticality

この拡張はクリティカルでない必要があります。

B.3.2. authorityKeyIdentifier

Authority Key Identifier 拡張機能は、証明書の署名に使用される秘密鍵に対応する公開鍵を識別します。この拡張機能は、CA 証明書が更新される場合など、発行者が複数の署名キーを持っている場合に役立ちます。

拡張機能は、次のいずれかまたは両方で設定されます。

- **keyIdentifier** フィールドに設定される明示的なキー識別子

- **authorityCertSerialNumber** フィールドで設定、シリアル番号、**authorityCertSerialNumber** フィールドで設定、証明書を識別。

keyIdentifier フィールドが存在する場合は、一致する **subjectKeyIdentifier** 拡張子を持つ証明書を選択するために使用されます。**AuthorityCertIssuer** および **authorityCertSerialNumber** フィールドが存在する場合、それらは **issuer** および **serialNumber** によって正しい証明書を識別するために使用されます。

この拡張子が存在しない場合は、発行者名のみが発行者証明書の識別に使用されます。

PKIX Part 1 では、自己署名ルート CA 証明書を除くすべての証明書にこの拡張機能が必要です。キー識別子が確立されていない場合、PKIX は **authorityCertIssuer** および **authorityCertSerialNumber** フィールドを指定することが推奨されます。これらのフィールドにより、発行元証明書の **authortiyCertIssuer** および **authorityCertSerialNumber** 拡張子で、発行者の証明書内の **SubjectName** フィールドおよび **CertificateSerialNumber** フィールドを照合して完全な証明書チェーンを作成できます。

OID

2.5.29.35

Criticality

この拡張は常に非クリティカルではなく、常に評価されます。

B.3.3. basicConstraints

この拡張機能は、証明書チェーンの検証プロセス中に、CA 証明書を識別し、証明書チェーンパスの長さの制約を適用するために使用されます。**cA** コンポーネントは、すべての CA 証明書に対して **true** に設定する必要があります。PKIX は、この拡張がエンドエンティティーの証明書に表示されないことを推奨します。

pathLenConstraint コンポーネントが存在する場合、その値は、エンドエンティティー証明書から始まり、チェーンを上に移動して、これまでに処理された CA 証明書の数よりも大きくなければなりません。**pathLenConstraint** を省略し、拡張機能が存在する場合は、チェーン内のすべての上位レベルの CA 証明書にこのコンポーネントを含めることはできません。

OID

2.5.29.19

Criticality

PKIX Part 1 では、この拡張が critical とマークされている必要があります。この拡張機能は、その重要度に関係なく評価されます。

B.3.4. certificatePoliciesExt

Certificate Policies 拡張機能は、1 つ以上のポリシーを定義します。各ポリシーは、OID と任意の修飾子で設定されます。拡張機能には、発行者の Certificate Practice Statement への URI を含めることも、ユーザー通知などの発行者情報をテキスト形式で埋め込むこともできます。この情報は、証明書が有効なアプリケーションで使用できます。

この拡張機能が存在する場合、PKIX Part 1 では、ポリシーを OID のみで識別するか、必要に応じて特定の推奨修飾子のみで識別することを推奨しています。

OID

2.5.29.32

Criticality

この拡張機能は、重要な場合と重要でない場合があります。

B.3.5. CRLDistributionPoints

この拡張は、CRL 情報の取得方法を定義します。システムが CRL 発行ポイントを使用するように設定されている場合は、これを使用する必要があります。

拡張機能に、タイプが URI に設定されている **DistributionPointName** が含まれている場合は、指定された失効理由のために現在の CRL へのポインターであると見なされ、**cRLIssuer** という名前で発行されます。URI の想定される値は、Subject Alternative Name 拡張に定義される値です。**distributionPoints** の理由を省略する場合は、すべての理由で CRL の取り消しを含める必要があります。**distributionPoint** が **cRLIssuer** を省略する場合、証明書を発行した CA によって CRL を発行する必要があります。

PKIX は、CA およびアプリケーションでこの拡張をサポートすることを推奨します。

OID

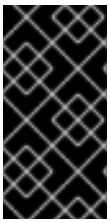
2.5.29.31

Criticality

PKIX では、この拡張機能を非クリティカルとマークし、すべての証明書でサポートすることが推奨されます。

B.3.6. extKeyUsage

Extended Key Usage 拡張機能は、認証された公開キーを使用できる目的を示します。これらの目的は、Key Usage 拡張機能に示されている基本的な目的に加えてまたはその代わりになる場合があります。



重要

キー使用法拡張と拡張キー使用法拡張の一貫性を維持する必要があります。詳細については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Key Usage および Extended Key Usage Consistency』 セクションを参照してください。

Extended Key Usage 拡張機能には、レスポンダーによって検証された証明書に署名した CA 署名キーが OCS P 署名キーでもない限り、OCSP レスポンダーの証明書内の **OCSP Signing** が含まれる必要があります。OCSP レスポンダーの証明書は、レスポンダーが検証する証明書に署名する CA によって直接発行される必要があります。

Key Usage、Extended Key Usage、および Basic Constraints 拡張機能は連携して機能し、証明書の使用目的を定義します。アプリケーションはこれらの拡張機能を使用して、不適切なコンテキストでの証明書の使用を禁止できます。

表B.36「PKIX Extended Key Usage Extension の使用」 は、この拡張機能に対して PKIX によって定義された用途を一覧表示します。表B.37「Private Extended Key Usage Extension の使用」 は、Netscape によって非公開で定義されたものを使用します。

OID

2.5.29.37

Criticality

この拡張機能がクリティカルとマークされている場合、証明書は指定された目的の 1 つにのみ使用する必要があります。クリティカルとマークされていない場合は、キーを識別するために使用できるアドバイザリーフィールドとして扱われますが、証明書の使用は指定された目的に制限されません。

表B.36 PKIX Extended Key Usage Extension の使用

使用	OID
サーバー認証	1.3.6.1.5.5.7.3.1
クライアント認証	1.3.6.1.5.5.7.3.2
コード署名	1.3.6.1.5.5.7.3.3
E メール	1.3.6.1.5.5.7.3.4
タイムスタンプ	1.3.6.1.5.5.7.3.8
OCSP 署名	1.3.6.1.5.5.7.3.9[a]

[a] OCSP 署名は PKIX Part 1 では定義されていませんが、RFC 2560 『X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP』 では定義されていません。

表B.37 Private Extended Key Usage Extension の使用

使用	OID
証明書トラストリスト署名	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 暗号化ファイルシステム	1.3.6.1.4.1.311.10.3.4
Netscape SGC	2.16.840.1.113730.4.1

B.3.7. issuerAltName 拡張

Issuer Alternative Name 拡張は、インターネットスタイルのアイデンティティーを証明書発行者に関連付けるために使用されます。名前には、Subject Alternative Name 拡張機能に定義したフォームを使用する必要があります。

OID

2.5.29.18

Criticality

PKIX Part 1 は、この拡張を非クリティカルとしてマークすることを推奨します。

B.3.8. keyUsage

Key Usage 拡張機能は、証明書に含まれるキーの目的を定義します。Key Usage、Extended Key Usage、および Basic Constraints 拡張機能は連携して機能し、証明書を使用できる目的を指定します。

重要

キー使用法拡張と拡張キー使用法拡張の一貫性を維持する必要があります。詳細については、『Red Hat Certificate System 9 Planning、Installation and Deployment Guide (Common Criteria Edition)』 の 『Key Usage および Extended Key Usage Consistency』 セクションを参照してください。

この拡張機能が全く含まれる場合は、以下のようにビットを設定します。

- TLS クライアント証明書、S/MIME 署名証明書、およびオブジェクト署名証明書用の **digitalSignature** (**0**)。
- 一部の S/MIME 署名証明書およびオブジェクト署名証明書用の **nonRepudiation** (**1**)

警告

このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。

- TLS サーバー証明書および S/MIME 暗号化証明書用の **keyEncipherment** (**2**)。
- **dataEncipherment** (**3**) サブジェクトの公開鍵を使用して、鍵情報ではなくユーザーデータを暗号化するとき。
- **keyAgreement** (**4**) サブジェクトの公開鍵がキー合意に使用される場合。
- すべての CA 署名証明書用の **keyCertSign** (**5**)。
- CRL の署名に使用される CA 署名証明書の **cRLSign** (**6**)。
- **encipherOnly** (**7**) 公開鍵がデータの暗号化にのみ使用される場合。このビットが設定されている場合 、**keyAgreement** も設定する必要があります。
- **decipherOnly** (**8**) 公開鍵がデータの暗号化にのみ使用される場合。このビットが設定されている場合 、**keyAgreement** も設定する必要があります。

表B.38「証明書の使用とそれに対応する鍵の使用方法」 は、通常の証明書使用ガイドラインをまとめています。

keyUsage 拡張機能が存在してクリティカルとマークされる場合は、証明書とキーの使用を強制するために使用されます。拡張機能は、キーの使用を制限するために使用されます。拡張機能が存在しないか非クリティカルな場合は、すべてのタイプの使用が許可されます。

keyUsage 拡張機能が存在する場合、クリティカルかどうかに関係なく、特定の操作に対して複数の証明書から選択するために使用されます。たとえば、操作用に個別の証明書とキーペアを持っているユーザーの個別の署名証明書と暗号化証明書を区別するために使用されます。

OID

2.5.29.15

Criticality

この拡張機能は、重要な場合と重要でない場合があります。PKIX Part 1 を使用する場合は、クリティカルなマークを付けることが推奨されます。

表B.38 証明書の使用とそれに対応する鍵の使用方法

証明書の目的	必要な鍵使用量のビット
CA 署名	• keyCertSign • cRLSign
TLS クライアント	digitalSignature
TLS サーバー	keyEncipherment
S/MIME 署名	digitalSignature
S/MIME 暗号化	keyEncipherment
証明書の署名	keyCertSign
オブジェクトの署名	digitalSignature

B.3.9. nameConstraints

この拡張機能は、CA 証明書でのみ使用でき、証明書パス内の後続の証明書のすべてのサブジェクト名を配置する必要がある名前空間を定義します。

OID

2.5.29.30

Criticality

PKIX Part 1 では、この拡張が critical とマークされている必要があります。

B.3.10. OCSPNocheck

拡張は OCSP 署名証明書に含まれることを目的としています。拡張機能は、(応答は OCSP レスポンダーによって再度署名され、クライアントは署名証明書の有効性ステータスを再度要求するため) OCSP レスポンダーに照会せずに署名証明書を信頼できることを OCSP クライアントに通知します。このエクステンションは null 値であり、その意味は存在するか、または存在しない場合により決定されます。

証明書にこの拡張機能が存在すると、OCSP クライアントはその証明書で署名された応答を信頼するため、この拡張機能の使用は慎重に管理する必要があります。OCSP 署名キーが危険にさらされると、証明書の有効期間中、PKI で証明書を検証するプロセス全体が危険にさらされます。したがって、**OCSPNocheck** を使用する証明書は短期間発行され、頻繁に更新される必要があります。

OID

1.3.6.1.5.5.7.48.4

Criticality

この拡張はクリティカルではありません。

B.3.11. policyConstraints

この拡張 (CA 証明書のみ) は、パスの検証を 2 つの方法で制限します。ポリシーマッピングを禁止したり、パス内の各証明書に受け入れ可能なポリシー識別子が含まれていることを要求したりするために使用できます。

PKIX は、(存在する場合)、この拡張は null シーケンスで設定されるべきではありません。利用可能な 2 つのフィールドが少なくとも 1 つ存在している必要があります。

OID

2.5.29.36

Criticality

この拡張機能は、重要な場合と重要でない場合があります。

B.3.12. policyMappings

Policy Mappings の拡張は、CA 証明書でのみ使用されます。これは、ある CA の対応するポリシーが別の CA のポリシーと同等であることを示すために使用される OID の 1 つ以上のペアをリストします。これは、ペア間の証明書のコンテキストで役に立つ場合があります。

この拡張機能は CA およびアプリケーションでサポートされる可能性があります。

OID

2.5.29.33

Criticality

この拡張はクリティカルでない必要があります。

B.3.13. privateKeyUsagePeriod

Private Key Usage Period の拡張機能により、証明書発行者は、証明書自体に秘密鍵に異なる有効期間を指定できます。この拡張は、デジタル署名鍵の使用を目的としています。

注記

PKIX Part 1 はこの拡張機能の使用に対して推奨されます。PKIX Part 1 に準拠する CA は、この拡張で証明書を生成 **することはできません**。

OID

2.5.29.16

B.3.14. subjectAltName

Subject Alternative Name 拡張には、CA によって認証された公開鍵にバインドされた ID の 1 つ以上の代替 (X.500 以外) 名が含まれます。証明書サブジェクト名に加えて、またはその代わりとして使用できます。定義された名前の形式には、インターネット電子メールアドレス (RFC-822 で定義された SMTP)、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、および URI (Uniform Resource Identifier) が含まれます。

PKIX では、サブジェクトフィールドで使用される X.500 識別名 (DN) 以外の名前形式で識別されるエンティティーに対してこの拡張機能が必要です。PKIX Part 1 では、この拡張機能とサブジェクトフィールドの関係に関する追加のルールを説明します。

電子メールアドレスは、Subject Alternative Name 拡張機能、証明書のサブジェクト名フィールド、またはその両方で指定できます。メールアドレスが件名の一部である場合は、PKCS #9 で定義された属性 **EmailAddress** の形式である必要があります。S/MIME をサポートするソフトウェアは、サブジェクト代替名拡張子またはサブジェクト名フィールドのいずれかから電子メールアドレスを読み取ることができる必要があります。

OID

2.5.29.17

Criticality

証明書の subject フィールドが空の場合は、この拡張機能にクリティカルのマークが付けられている必要があります。

B.3.15. subjectDirectoryAttributes

Subject Directory Attributes 機能拡張は、証明書の件名に必要なディレクトリー属性の値をすべて伝えます。提案されている PKIX 標準の必須部分としては推奨されていませんが、ローカル環境で使用できます。

OID

2.5.29.9

Criticality

PKIX Part 1 では、この拡張が非クリティカルなものとしてマークされている必要があります。

B.3.16. subjectKeyIdentifier

Subject Key Identifier 拡張は、この証明書で認定された公開鍵を識別します。この拡張機能は、特定のサブジェクト名に複数の公開鍵が使用可能な場合に、公開鍵を区別する方法を提供します。

この拡張機能の値は、PKIX で推奨されているとおり、証明書の DER エンコードされた **subjectPublicKey** の SHA-1 ハッシュを実行して計算する必要があります、Subject Key Identifier 拡張機能は、CA 証明書の Authority Key Identifier 拡張機能と組み合わせて使用されます。CA 証明書に Subject Key Identifier 拡張がある場合、検証される証明書の Authority Key Identifier のキー識別子は、CA の Subject Key Identifier 拡張のキー識別子と一致する必要があります。この場合、検証者がキー識別子を再計算する必要はありません。

PKIX Part 1 では、すべての CA 証明書にこの拡張機能が必要であり、他のすべての証明書にこの拡張機能を推奨しています。

OID

2.5.29.14

Criticality

この拡張機能は常にクリティカルではありません。

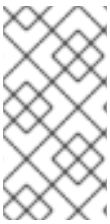
B.4. CRL 拡張機能

B.4.1. CRL 拡張について

最初の発行以来、CRL 形式の X.509 標準が修正され、CRL 内に追加情報が含まれるようになりました。この情報は CRL 拡張機能により追加されます。

ANSI X9 および ISO/IEC/ITU for X.509 CRLs [X.509] [X9.55] で定義されている拡張機能により、追加の属性を CRL に関連付けることができます。RFC5280 で入手可能な『Internet X.509 Public Key Infrastructure Certificate and CRL Profile』 は、CRL で使用される拡張機能のセットを推奨しています。これらの拡張機能は *標準 CRL 拡張機能* と呼ばれます。

この規格では、カスタム拡張機能を作成して CRL に含めることもできます。これらの拡張機能は、*プライベート* 拡張機能、*プロプライエタリー* 拡張機能、または *カスタム* CRL 拡張機能と呼ばれ、組織またはビジネスに固有の情報を伝達します。アプリケーションは、プライベートの重要な拡張機能を含む CRL を検証できない場合があるため、一般的なコンテキストでカスタム拡張機能を使用することはお勧めしません。



注記

Abstract Syntax Notation One (ASN.1) および Distinguished Encoding Rules (DER) 標準は、CCITT 勧告 X.208 および X.209 で指定されています。ASN.1 および DER の概要については、RSA Laboratories の Web サイト (http://www.rsa.com) で入手できる『A Layman's Guide to a Subset of ASN.1, BER, and DER』を参照してください。

B.4.1.1. CRL 拡張の構造

CRL 拡張機能は、以下の部分で設定されます。

- 拡張のオブジェクト識別子 (OID)。この識別子は、拡張子を一意に識別します。また、値フィールドの ASN.1 タイプの値や、値が解釈される方法も決定します。拡張機能が CRL に表示されると、OID は拡張機能 ID フィールド (**extnID**) として示されます。また、対応する ASN.1 エンコード構造は、オクテット文字列の値 (**extnValue**) として示されます。例は 例 B.4「Pretty-Print 証明書拡張機能の例」 で示されます。

- **critical** というフラグまたはブール値フィールド

  このフィールドに割り当てられた **true** 値または **false** 値は、拡張機能が CRL にとってクリティカルか非クリティカルかを示します。

  - 拡張機能が重要であり、拡張機能の ID に基づいて拡張機能を理解しないアプリケーションに CRL が送信された場合は、アプリケーションが CRL を拒否する必要があります。

  - 拡張機能が重要ではなく、拡張機能の ID に基づいて拡張機能を理解しないアプリケーションに CRL が送信された場合、アプリケーションは拡張機能を無視して CRL を受け入れることができます。

- 拡張の値の DER エンコーディングを含む octet 文字列。

CRL を受信するアプリケーションは、拡張 ID を確認して、ID を認識できるかどうかを判断します。可能であれば、エクステンション ID を使用して、使用する値のタイプを判断します。

B.4.1.2. CRL および CRL エントリー拡張のサンプル

以下は、X.509 CRL バージョン 2 の拡張機能の例です。Certificate System は、以下に示すように、読み取り可能な pretty-print 形式で CRL を表示できます。例に示されているように、CRL 拡張は順番に表示され、特定の拡張の 1 つのインスタンスのみが CRL ごとに表示される場合があります。たとえば、CRL に含まれる Authority Key Identifier 拡張機能は 1 つだけです。ただし、CRL-entry 拡張は CRL の適切なエントリーに表示されます。

```
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
```

デルタ CRL は、最後の CRL が公開されてからの変更のみを含む CRL のサブセットです。デルタ CRL インジケーター拡張を含む CRL はデルタ CRL です。

```
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. 標準 X.509 v3 CRL 拡張機能リファレンス

証明書の拡張に加えて、X.509 で提案されている標準では、CRL の拡張が定義されており、追加の属性をインターネット CRL に関連付ける方法が提供されています。これらは、CRL 自体の拡張と、CRL の個々の証明書エントリーの拡張の 2 種類のうちの 1 つです。

- 「CRL の拡張機能」
- 「CRL エントリー拡張」

B.4.2.1. CRL の拡張機能

以下の CRL の説明は、インターネット X.509 v3 公開鍵インフラストラクチャーが提案する標準の一部として定義されています。

- 「authorityInfoAccess」
- 「authorityKeyIdentifier」
- 「CRLNumber」
- 「deltaCRLIndicator」
- 「FreshestCRL」
- 「issuerAltName」
- 「issuingDistributionPoint」
- 「FreshestCRL」

B.4.2.1.1. authorityInfoAccess

Authority Information Access 拡張は、デルタ CRL 情報の取得方法を識別します。freshestCRL 拡張機能は完全な CRL に配置され、最新のデルタ CRL の場所を示します。

OID

1.3.6.1.5.5.7.1.1

Criticality

PKIX では、この拡張は重要ではありません。

パラメーター

表B.39 認証局情報アクセス設定パラメーター

パラメーター	説明
enable	ルールを有効または無効するかどうかを指定します。デフォルトでは、この拡張機能は無効になっています。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。
numberOfAccessDescriptions	0 から任意の正の整数までのアクセス記述の数を示します。デフォルトは 0 です。 このパラメーターを 0 以外の整数に設定する場合は、数値を設定し、OK をクリックしてウィンドウを閉じます。ルールの編集ウィンドウを再度開き、ポイントを設定するフィールドが存在します。
accessMethod**n**	このパラメーターで許可される値は calssuers のみです。calssuers メソッドは、利用可能な情報に CRL の署名の検証に使用できる証明書がリストされている場合に使用されます。AIA 拡張が CRL に含まれる場合、他の方法は使用しないでください。
accessLocationType**n**	**n** アクセスの説明のアクセス場所を指定します。オプションは **DirectoryName** または **URI** です。
accessLocation**n**	**accessLocationType** が **DirectoryName** に設定されている場合、値は証明書のサブジェクト名と同様に、X.500 名の形式の文字列である必要があります。たとえば、 **CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 **accessLocationType** が **URI** に設定されている場合、名前は URI である必要があります。URI は絶対パス名で、ホストを指定する必要があります。例: **http://testCA.example.com/get/crls/here/**。

B.4.2.1.2. authorityKeyIdentifier

CRL の AuthorityKey Identifier 拡張機能は、CRL の署名に使用される秘密鍵に対応する公開鍵を識別します。詳細は、証明書エクステンション(「authorityKeyIdentifier」)を参照してください。

PKIX 標準では、CA の公開鍵は、たとえば鍵が更新されたときに変更される可能性があるため、または複数の同時鍵ペアまたは鍵切り替えのために CA が複数の署名鍵を持つ可能性があるため、CA が発行するすべての CRL にこの拡張機能を含める必要があることを推奨しています。このような場合、CA は複数のキーペアで終了します。証明書の署名を検証する場合、他のアプリケーションは、署名で使用されたキーを知る必要があります。

OID

2.5.29.35

パラメーター

表B.40 AuthorityKeyIdentifierExt 設定パラメーター

パラメーター	説明
enable	ルールを有効または無効するかどうかを指定します。デフォルトでは、この拡張機能は無効になっています。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。

B.4.2.1.3. CRLNumber

CRLNumber 拡張は、CA が発行する各 CRL の連続する番号を指定します。ユーザーは、特定の CRL が別の CRL を上書きするタイミングを簡単に判断できます。PKIX では、すべての CRL にこの拡張が必要です。

OID

2.5.29.20

Criticality

この拡張は重要ではありません。

パラメーター

表B.41 CRLNumber 設定パラメーター

パラメーター	説明
enable	ルールが有効であるかどうかを指定します (デフォルト)。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。

B.4.2.1.4. deltaCRLIndicator

deltaCRLIndicator 拡張機能は、デルタ CRL を生成します。これは、最後の CRL 以降に取り消された証明書のみのリストです。また、ベース CRL への参照も含まれています。これにより、ローカルデータベースに既に存在する未変更の情報を無視しながら、ローカルデータベースが更新されます。これにより、失効情報を CRL 構造以外の形式で格納するアプリケーションの処理時間を大幅に改善できます。

OID

2.5.29.27

Criticality

PKIX では、その拡張が存在する場合はこの拡張が必須でなければなりません。

パラメーター

表B.42 DeltaCRL 設定パラメーター

パラメーター	説明
enable	ルールが有効かどうかを指定します。デフォルトでは無効になっています。
critical	拡張機能がクリティカル、または非クリティカルを設定します。デフォルトでは、これはクリティカルになっています。

B.4.2.1.5. FreshestCRL

freshestCRL 拡張機能は、デルタ CRL 情報の取得方法を識別します。freshestCRL 拡張機能は完全な CRL に配置され、最新のデルタ CRL の場所を示します。

OID

2.5.29.46

Criticality

PKIX では、この拡張機能は非クリティカルである必要があります。

パラメーター

表B.43 FreshestCRL 設定パラメーター

パラメーター	説明
enable	拡張機能ルールが有効かどうかを指定します。デフォルトでは、これは無効になっています。
critical	拡張にクリティカルまたは非クリティカルのマークを付けます。デフォルトはクリティカルではありません。
numPoints	デルタ CRL の発行ポイントの数を、**0** から任意の正の整数に指定します。デフォルトは **0** です。これを 0 以外の整数に設定する場合は、数値を設定してから **OK** をクリックしてウィンドウを閉じます。ルールの編集ウィンドウを再度開き、これらのポイントを設定するフィールドが存在するようになります。

パラメーター	説明
pointType**n**	**n** 発行ポイントの発行ポイントのタイプを指定します。**numPoints** で指定する数字ごとに、**pointType** パラメーターの等しい数があります。オプションは **DirectoryName** または **URIName** です。
pointName**n**	**pointType** が **directoryName** に設定されている場合、値は証明書のサブジェクト名と同様に、X.500 名の形式の文字列である必要があります。たとえば、**CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 **pointType** が **URIName** に設定されている場合、名前は URI である必要があります。URI は絶対パス名で、ホストを指定する必要があります。例: **http://testCA.example.com/get/crls/here/**。

B.4.2.1.6. issuerAltName

Issuer Alternative Name 拡張機能を使用すると、メールアドレス、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、URI (Uniform Resource Indicator) などのバインディング属性など、CRL の発行者を使用して、追加の ID を CRL の発行者に関連付けることができます。詳細は、証明書エクステンション(「issuerAltName 拡張」)を参照してください。

OID

2.5.29.18

パラメーター

表B.44 IssuerAlternativeName 設定パラメーター

パラメーター	説明
enable	拡張ルールが有効であるかどうかを設定します。デフォルトでは無効になっています。
critical	拡張がクリティカルかどうかを設定します。デフォルトでは、これは非クリティカルになります。

パラメーター	説明
numNames	拡張で許可される代替名または ID の合計数を設定します。それぞれの名前には、設定パラメーター (**nameType** および **name**) のセットがあります。これには適切な値が必要です。そうでない場合、ルールはエラーを返します。このフィールドで指定された値を変更して、ID の総数を変更します。拡張機能に含めることができる ID の総数に制限はありません。設定パラメーターの各セットは、このフィールドの値から派生した整数によって区別されます。たとえば、**numNames** パラメーターが **2** に設定されている場合、派生整数は **0** と **1** です。
nameType**n**	general-name タイプを指定します。次のいずれかになります。 ● 名前がインターネットメールアドレスの場合は **rfc822Name**。 ● 名前が X.500 ディレクトリー名である場合に **directoryName**。 ● **dNSName** 名前が DNS 名である場合。 ● **ediPartyName** 名前が EDI 当 事名である場合。 ● **URL** 名前が URI (デフォルト) の場合。 ● **iPAddress** 名前が IP アドレスの場合。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 ● 名前がオブジェクト識別子である場合は **OID**。 ● 名前が他の名前形式である場合は **otherName** です。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。
name**n**	一般名の値を指定します。許可される値は、**nameType** フィールドで指定された名前の種類によって異なります。

パラメーター	説明
	• **rfc822Name** の場合は、値が **local-part@domain** 形式の有効なインターネットメールアドレスである必要があります。 • **directoryName** の場合、この値は証明書のサブジェクト名と同様に X.500 名である必要があります。たとえば、**CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 • **dNSName** の場合、値は DNS 形式の有効なドメイン名である必要があります。例: **testCA.example.com** • **eDIPartyName** の場合、名前は IA5String である必要があります。例: **Example Corporation**。 • **URL** の場合、値は相対的でない URI である必要があります。例: **http://testCA.example.com**。 • **iPAddress** の場合、値は、ドットで区切られたコンポーネント表記で指定された有効な IP アドレスである必要があります。IP アドレス、またはネットマスクを含む IP アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 • **OID** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。カスタム OID を使用してサーバーを評価およびテストできますが、実稼働環境では、OID の定義および ID のサブツリーの登録に関する ISO 規則に準拠します。 • **otherName** の場合、名前には他の形式が使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any** は、**/var/lib/pki/pki-tomcat/ca/othername.txt** などのサブツリーをしている base-64 エンコードファイルに文字列を設定します。**KerberosName** には、**realm1|0|userID1,userID2** などの **Realm|NameType|NameStrings** 形式になります。名前は、base-64 でエンコードされた形式の一般名を含むファイルへの絶対

パラメーター	説明
	パスである必要があります。たとえば、/var/lib/pki/pki-tomcat/ca/extn/ian/othername.txt になります。

B.4.2.1.7. issuingDistributionPoint

Issuing Distribution Point CRL 拡張機能は、特定の CRL の CRL ディストリビューションポイントを識別し、エンドエンティティー証明書のみの失効、CA 証明書のみ、または理由コードのセットが制限されている失効証明書など、対象となる失効の種類を示します。

PKIX Part I ではこの拡張は必要ありません。

OID

2.5.29.28

Criticality

PKIX では、その拡張が存在する場合はこの拡張が必須でなければなりません。

パラメーター

表B.45 IssuingDistributionPoint 設定パラメーター

パラメーター	説明
enable	拡張機能を有効にするかどうかを設定します。デフォルトは無効です。
critical	拡張機能をクリティカル、デフォルト、または非クリティカルとしてマークします。
pointType	以下から発行配布ポイントのタイプを指定します。 ● **directoryName** は、タイプが X.500 ディレクトリー名であることを指定します。 ● **URI** は、タイプが統一されたリソースインジケーターであることを指定します。 ● **RelativeToIssuer** は、タイプが **ou=Engineering** などの DN の単一ノードを表す相対識別名 (RDN) であることを指定します。

パラメーター	説明
pointName	発行されたディストリビューションポイントの名前を指定します。分散ポイントの名前は、**pointType** パラメーターに指定された値によって異なります。 ● **directoryName** の場合、名前は X.500 名である必要があります。たとえば、**cn=CRLCentral,ou=Research Dept,o=Example Corporation,c=US** となります。 ● **URIName** では、名前は絶対パス名であり、ホストを指定する URI である必要があります。例: **http://testCA.example.com/get/crls/here/**。  **注記** CRL は、CRL 発行ポイントに対応するディレクトリーエントリーに格納される場合があります。これは、CA のディレクトリーエントリーとは異なる場合があります。
onlySomeReasons	ディストリビューションポイントに関連付けられた理由コードを指定します。 許容値は、理由コード (**unspecified**、**keyCompromise**、**cACompromise**、**affiliationChanged**、**asuperseded**、**cessationOfOperation**、**certificateHold**、および **removeFromCRL**) の組み合わせです。ディストリビューションポイントにすべての理由コード (デフォルト) を含む失効した証明書が含まれている場合は、フィールドを空白のままにします。
onlyContainsCACerts	設定されている場合に限り、ディストリビューションポイントにユーザー証明書が含まれることを指定します。デフォルトでは、これは設定されていません。つまり、ディストリビューションポイントにはすべての種類の証明書が含まれています。
indirectCRL	ディストリビューションポイントに間接 CRL が含まれていることを指定します。デフォルトでは選択されていません。

B.4.2.2. CRL エントリー拡張

次のセクションでは、インターネット X.509v3 公開鍵インフラストラクチャーで提案されている標準の一部として定義されている CRL エントリー拡張タイプを示します。これらの拡張機能はすべてクリティカルではありません。

B.4.2.2.1. certificateIssuer

Certificate Issuer 拡張機能は、間接 CRL のエントリーに関連付けられている証明書発行者を識別します。

この拡張機能は、Certificate System でサポートされていない間接 CRL でのみ使用されます。

OID

2.5.29.29

B.4.2.2.2. invalidityDate

Invalidity Date 拡張機能は、秘密鍵が侵害された日付、または証明書が無効になった日付を提供します。

OID

2.5.29.24

パラメーター

表B.46 InvalidityDate 設定パラメーター

パラメーター	説明
enable	拡張機能ルールを有効にするか無効にするかを設定します。デフォルトでは、これは有効になります。
critical	拡張機能をクリティカルまたは非クリティカルとしてマークします。デフォルトでは、これは非クリティカルではありません。

B.4.2.2.3. CRLReason

Reason Code 拡張は、証明書失効リストの理由を識別します。

OID

2.5.29.21

パラメーター

表B.47 CRLReason 設定パラメーター

パラメーター	説明
enable	拡張機能ルールを有効にするか無効にするかを設定します。デフォルトでは、これは有効になります。

パラメーター	説明
critical	拡張にクリティカルまたは非クリティカルのマークを付けます。デフォルトでは、これはクリティカルではありません。

B.4.3. Netscape で定義された証明書拡張のリファレンス

Netscape は、その製品に対して特定の証明書拡張を定義しました。一部の拡張機能は現在廃止されており、その他の拡張機能は X.509 提案標準で定義されている拡張機能に取って代わられています。すべての Netscape 拡張機能は、証明書に存在することでその証明書が他のクライアントと互換性がなくなることがないように、非クリティカルなものとしてタグ付けする必要があります。

B.4.3.1. netscape-cert-type

Netscape Certificate Type 拡張機能を使用して、証明書を使用できる目的を制限できます。X.509 v3 拡張 「extKeyUsage」 および 「basicConstraints」 に置き換えられました。

拡張機能が証明書に存在する場合は、指定した使用に対して証明書を制限します。拡張機能が存在しない場合、証明書はオブジェクト署名を除くすべてのアプリケーションで使用できます。

値はビット文字列であり、個々のビット位置が設定されると、次のように特定の用途のために証明書を認証します。

- ビット 0: TLS クライアント証明書
- ビット 1: TLS サーバー証明書
- ビット 2 - S/MIME 証明書
- ビット 3 - オブジェクト署名証明書
- ビット 4 - 予約
- ビット 5: TLS CA 証明書
- ビット 6 - S/MIME CA 証明書
- ビット 7 - オブジェクト署名 CA 証明書

OID

2.16.840.1.113730.1.1

B.4.3.2. netscape-comment

この拡張機能の値は IA5String です。これは、証明書を表示する際にユーザーに表示されるコメントです。

OID

2.16.840.1.113730.13

付録C 公開モジュールのリファレンス

いくつかのパブリッシャー、マッパー、およびルールモジュールは、デフォルトで Certificate Manager を使用して設定されます。

- 「パブリッシャープラグインモジュール」
- 「マッパープラグインモジュール 」
- 「ルールインスタンス」

C.1. パブリッシャープラグインモジュール

このセクションでは、Certificate Manager に提供されるパブリッシャーモジュールを説明します。これらのモジュールは Certificate Manager で使用され、特定のパブリッシャーインスタンスを有効および設定します。

- 「FileBasedPublisher」
- 「LdapCaCertPublisher」
- 「LdapUserCertPublisher」
- 「LdapCrlPublisher」
- 「LdapDeltaCrlPublisher」
- 「LdapCertificatePairPublisher」
- 「OCSPPublisher」

C.1.1. FileBasedPublisher

FileBasedPublisher プラグインモジュールは、証明書および CRL をファイルに公開するように Certificate Manager を設定します。このプラグインは、発行元の設定時に選択したチェックボックスに応じて、base-64 でエンコードされたファイル、DER でエンコードされたファイル、またはその両方を発行できます。証明書および CRL の内容は、**PrettyPrintCert** および **PrettyPrintCRL** ツールを使用してファイルを変換して表示できます。base-64 および DER でエンコードされた証明書および CRL でコンテンツを表示する方法は、「ファイルに公開される証明書および CRL の表示」 を参照してください。

デフォルトでは、Certificate Manager は **FileBasedPublisher** モジュールのインスタンスを作成しません。

表C.1 FileBasedPublisher 設定パラメーター

パラメーター	詳細
Publisher ID	パブリッシャーの名前、スペースを含まない英数字の文字列を指定します。例: **PublishCertsToFile**

パラメーター	詳細
directory	Certificate Manager がファイルを作成するディレクトリーへの完全なパスを指定します。パスは絶対パスにすることも、Certificate System インスタンスディレクトリーからの相対パスにすることもできます。たとえば、**/export/CS/certificates** です。

C.1.2. LdapCaCertPublisher

LdapCaCertPublisher プラグインモジュールは、CA 証明書の CA 証明書を CA のディレクトリーエントリーの **caCertificate;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

モジュールは、CA エントリーのオブジェクトクラスを pkiCA または certificationAuthority に変換していない場合はこれを **pkiCA** または **certificationAuthority** に変換します。同様に、CA に他の証明書がない場合に、**pkiCA** または **certificationAuthority** オブジェクトクラスも削除します。

インストール中に、Certificate Manager は、CA 証明書をディレクトリーに発行するための **LdapCaCertPublisher** モジュールのインスタンスを自動的に作成します。

表C.2 LdapCaCertPublisher 設定パラメーター

パラメーター	詳細
caCertAttr	CA 証明書を公開する LDAP ディレクトリー属性を指定します。これは、**caCertificate;binary** でなければなりません。
caObjectClass	ディレクトリー内の CA のエントリーのオブジェクトクラスを指定します。これは **pkiCA** または **certificationAuthority** でなければなりません。

C.1.3. LdapUserCertPublisher

LdapUserCertPublisher プラグインモジュールは、User 証明書の User 証明書を ユーザーのディレクトリーエントリーの **userCertificate;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

このモジュールは、エンドエンティティー証明書を LDAP ディレクトリーに公開するために使用されます。エンドエンティティー証明書の種類には、TLS クライアント、S/MIME、TLS サーバー、および OCSP レスポンダーが含まれます。

インストール中、Certificate Manager は、ディレクトリーにエンドエンティティー証明書を発行するための **LdapUserCertPublisher** モジュールのインスタンスを自動的に作成します。

表C.3 LdapUserCertPublisher 設定パラメーター

パラメーター	詳細

パラメーター	詳細
certAttr	Certificate Manager が証明書を公開するマップされたエントリーのディレクトリー属性を指定します。これは **userCertificate;binary** である必要があります。

C.1.4. LdapCrlPublisher

LdapCrlPublisher プラグインモジュールは、CRL をディレクトリーエントリーの **certificateRevocationList;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

インストール中に、Certificate Manage は、自動的に CA 証明書をディレクトリーに公開するための **LdapCrlPublisher** モジュールのインスタンスを作成します。

表C.4 LdapCrlPublisher 設定パラメーター

パラメーター	詳細
crlAttr	証明書マネージャーが CRL を公開するマップされたエントリーのディレクトリー属性を指定します。これは **certificateRevocationList;binary** でなければなりません。

C.1.5. LdapDeltaCrlPublisher

LdapDeltaCrlPublisher プラグインモジュールは、Delta CRL をディレクトリーエントリーの **deltaRevocationList** 属性に公開または公開しないように、証明書マネージャーを設定します。

インストール中に、Certificate Manage は、自動的に CA 証明書をディレクトリーに公開するための **LdapDeltaCrlPublisher** モジュールのインスタンスを作成します。

表C.5 LdapDeltaCrlPublisher 設定パラメーター

パラメーター	詳細
crlAttr	Certificate Manager がデルタ CRL を公開するマップされたエントリーのディレクトリー属性を指定します。これは **deltaRevocationList;binary** である必要があります。

C.1.6. LdapCertificatePairPublisher

LdapCertificatePairPublisher プラグインモジュールは、CA のディレクトリーエントリーの **crossCertPair;binary** 属性に、クロス署名証明書を公開または公開しないように証明書マネージャーを設定します。

モジュールは、CA エントリーのオブジェクトクラスを pkiCA または certificationAuthority に変換していない場合はこれを **pkiCA** または **certificationAuthority** に変換します。同様に、CA に他の証明書がない場合に、**pkiCA** または **certificationAuthority** オブジェクトクラスも削除します。

インストール時に、証明書マネージャーは **LdapCrossCertPairPublisher** という名前の **LdapCertificatePairPublisher** モジュールのインスタンスを作成し、複数の署名された証明書をディレクトリーに公開します。

表C.6 LdapCertificatePairPublisher パラメーター

パラメーター	詳細
crossCertPairAttr	CA 証明書を公開する LDAP ディレクトリー属性を指定します。これは **crossCertificatePair;binary** でなければなりません。
caObjectClass	ディレクトリー内の CA のエントリーのオブジェクトクラスを指定します。これは **pkiCA** または **certificationAuthority** でなければなりません。

C.1.7. OCSPPublisher

OCSPPublisher プラグインモジュールは、CRL を Online Certificate Status Manager に公開するように Certificate Manager を設定します。

Certificate Manager は、インストール時に **OCSPPublisher** モジュールのインスタンスを作成しません。

表C.7 OCSPPublisher パラメーター

パラメーター	詳細
host	Online Certificate Status Manager の完全修飾ホスト名を指定します。
ポート	Online Certificate Status Manager が Certificate Manager をリッスンしているポート番号を指定します。これは、オンライン証明書ステータスマネージャーの TLS ポート番号です。
path	CRL を公開するためのパスを指定します。これはデフォルトのパス **/ocsp/agent/ocsp/addCRL** でなければなりません。
enableClientAuth	クライアント (証明書ベースの) 認証を使用して OCSP サービスにアクセスするかどうかを設定します。
ニックネーム	クライアント認証に使用する OCSP サービスのデータベース内の証明書のニックネームを指定します。これは、**enableClientAuth** オプションが true に設定されている場合にのみ使用されます。

C.2. マッパープラグインモジュール

このセクションでは、Certificate Manager に提供されるマッパープラグインモジュールを説明します。これらのモジュールは、特定のマッパーインスタンスを有効化および設定するように Certificate Manager を設定します。

利用可能なマッパープラグインモジュールには以下が含まれます。

- 「LdapCaSimpleMap」
- 「LdapDNExactMap」
- 「LdapSimpleMap」
- 「LdapSubjAttrMap」
- 「LdapDNCompsMap」

C.2.1. LdapCaSimpleMap

LdapCaSimpleMap プラグインモジュールは、LDAP ディレクトリーに CA のエントリーを自動的に作成し、証明書要求、証明書サブジェクト名、証明書拡張子で指定されたコンポーネント、および属性変数アサーション (AVA) 定数からエントリーの DN を作成することにより、CA の証明書をディレクトリーエントリーにマップするように Certificate Manager を設定します。AVA の詳細は、ディレクトリーのドキュメントを参照してください。

CA 証明書マッパーは、CA のエントリーを作成するか、証明書を既存のエントリーにマップするか、またはその両方を行うかを指定します。

公開ディレクトリーに CA エントリーがすでに存在し、このマッパーの **dnPattern** パラメーターに割り当てられた値が変更しても、**uid** 属性および **o** 属性が同じである場合、マッパーは 2 つ目の CA エントリーの作成に失敗します。たとえば、ディレクトリーに **uid=CA,ou=Marketing,o=example.com** の CA エントリーがすでにあり、**uid=CA,ou=Engineering,o=example.com** で別の CA エントリーを作成するようマッパーが設定されている場合、操作は失敗します。

ディレクトリーの *UID Uniqueness* プラグインが特定のベース DN に設定されているため、操作が失敗する場合があります。この設定により、ディレクトリーがそのベース DN の下に同じ UID を持つ 2 つのエントリーを持つことを防ぎます。この例では、ディレクトリーが **o=example.com** の下に同じ UID **CA** を持つ 2 つのエントリーを持つことを防ぎます。

マッパーが 2 番目の CA エントリーの作成に失敗した場合は、UID 一意性プラグインが設定されているベース DN を確認し、同じ UID のエントリーがディレクトリーにすでに存在するかどうかを確認します。必要に応じて、マッパー設定を調整するか、古い CA エントリーを削除するか、プラグインをコメントアウトするか、エントリーを手動で作成します。

インストール時に、Certificate Manager は CA 証明書マッパーモジュールの 2 つのインスタンスを自動的に作成します。マッパーの名前は以下のように命名されます。

- CRL の **LdapCrlMap** (「LdapCrlMap」 を参照)
- CA の **LdapCaCertMap** (「LdapCaCertMap」 を参照)

表C.8 LdapCaSimpleMap 設定パラメーター

パラメーター	詳細

パラメーター	詳細
createCAEntry	選択した場合は CA のエントリーを作成します (デフォルト)。 選択した場合、Certificate Manager は最初にディレクトリーに CA のエントリーを作成しようとします。Certificate Manager がエントリーの作成に成功すると、CA の証明書をエントリーに公開しようとします。このチェックボックスを選択しないと、公開するためにエントリーがすでに存在している必要があります。
dnPattern	Certificate Manager が公開ディレクトリーで CA のエントリーを検索するために構築するために使用する DN パターンを指定します。**dnPattern** の値は、コンマで区切られた AVAs の一覧です。AVA は、Certificate Manager が、(**cn=$subj.cn** など) の証明書のサブジェクト名または制約から派生する変数 (**o=Example Corporation** など) です。 CA 証明書に **cn** コンポーネントがない場合に、サブジェクト名のコンポーネントで、CA 証明書マッピングの DN パターンを調整して、CA 証明書が公開されるディレクトリー内のエントリーの DN を反映します。たとえば、CA 証明書のサブジェクト DN が **o=Example Corporation** で、ディレクトリーの CA エントリーが **cn=Certificate Authority, o=Example Corporation** の場合、パターンは **cn=Certificate Authority, o=$subj.o** になります。 ● 例 1: **uid=CertMgr, o=Example Corporation** ● 例 2: **cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US** ● 例 3: **uid=$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou** 上記の例では、**$req** は証明書要求の属性、**$subj** は証明書のサブジェクト名の属性、**$ext** は証明書の拡張子の属性を取ります。

C.2.1.1. LdapCaCertMap

LdapCaCertMap マッパーは、**LdapCaSimpleMap** モジュールのインスタンスです。Certificate Manager は、インストール時にこのマッパーを自動的に作成します。

このマッパーは、ディレクトリーに CA のエントリーを作成し、CA 証明書をディレクトリー内の CA のエントリーを作成します。

デフォルトでは、マッパーはディレクトリーに CA のエントリーを作成するように設定されています。CA のエントリーを見つけるためのデフォルトの DN パターンは次のとおりです。

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.1.2. LdapCrlMap

LdapCrlMap マッパーは **LdapCaSimpleMap** モジュールのインスタンスです。Certificate Manager は、インストール時にこのマッパーを自動的に作成します。

このマッパーは、ディレクトリー内に CA のエントリーを作成し、CRL をディレクトリー内の CA のエントリーにマップします。

デフォルトでは、マッパーは、ディレクトリーに CA のエントリーを作成するように設定されます。CA のエントリーを見つけるデフォルトの DN パターンは以下のとおりです。

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.2. LdapDNExactMap

LdapDNExactMap プラグインモジュールは、証明書のサブジェクト名と一致する LDAP エントリー DN を検索することにより、証明書を LDAP ディレクトリーエントリーにマップするように Certificate Manager を設定します。このマッパーを使用するには、各証明書のサブジェクト名がディレクトリーエントリーの DN と完全に一致する必要があります。たとえば、証明書サブジェクト名が **uid=jdoe, o=Example Corporation, c=US** の場合には、証明書マネージャーは **DN uid=jdoe, o=Example Corporation, c=US** を持つエントリーのみを検索します。

一致するエントリーが見つからない場合、サーバーはエラーを返し、証明書を公開しません。

このマッパーは、証明書からすべての値を取得するため、パラメーターに値を必要としません。

C.2.3. LdapSimpleMap

LdapSimpleMap プラグインモジュールは、証明書要求、証明書のサブジェクト名、証明書の拡張子、および属性変数アサーション (AVA) 定数で指定されたコンポーネントからエントリーの DN を引き出すことで、証明書を LDAP ディレクトリーエントリーにマッピングするように証明書マネージャーを設定します。AVA の詳細は、ディレクトリーのドキュメントを参照してください。

デフォルトでは、Certificate Manager は単純なマッパーに基づくマッパールールを使用します。インストール中に、Certificate Manager は、**LdapUserCertMap** という名前が付けられた単純なマッパーモジュールのインスタンスを自動的に作成します。デフォルトのマッパーは、さまざまなタイプのエンドエンティティー証明書を対応するディレクトリーエントリーにマップします。

シンプルなマッパーには 1 つのパラメーター **dnPattern** が必要です。**dnPattern** の値は、コンマで区切られた AVAs の一覧です。AVA は、**uid=$subj.UID** などの変数や、**o=Example Corporation** などの定数になります。

- 例 1: **uid=CertMgr, o=Example Corporation**
- 例 2: **cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US**
- 例 3: uid=**$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou**

例では、**$req** は、証明書要求から属性を取得し、**$subj** は、証明書のサブジェクト名から属性を取得し、**$ext** は、証明書拡張から属性を取得します。

C.2.4. LdapSubjAttrMap

LdapSubjAttrMap プラグインモジュールは、設定可能な LDAP 属性を使用して証明書を LDAP ディレクトリーエントリーにマップするように Certificate Manager を設定します。このマッパーを使用するには、ディレクトリーエントリーに指定された LDAP 属性を含める必要があります。

Certificate Manager は、サブジェクト DN 全体と完全に一致する値を持つ属性をディレクトリーで検索するため、このマッパーにはサブジェクト DN の正確なパターンが必要です。たとえば、指定された LDAP 属性が **certSubjectDN** で、証明書サブジェクト名が **uid=jdoe, o=Example Corporation, c=US** の場合、証明書マネージャーは、**certSubjectDN=uid=jdoe** 属性のあるエントリーに対してディレクトリーを検索します。

一致するエントリーが見つからない場合、サーバーはエラーを返し、ログに書き込みます。

表C.9「LdapSubjAttrMap Parameters」 これらのパラメーターを説明します。

表C.9 LdapSubjAttrMap Parameters

パラメーター	詳細
certSubjNameAttr	証明書のサブジェクト名を値として含む LDAP 属性の名前を指定します。デフォルトは **certSubjectName** ですが、任意の LDAP 属性に設定できます。
searchBase	属性検索を開始するベース DN を指定します。許容値は、**o=example.com, c=US** などの LDAP エントリーの有効な DN です。

C.2.5. LdapDNCompsMap

LdapDNCompsMap プラグインモジュールは、DN コンポーネントマッパーを実装します。このマッパーは、証明書のサブジェクト名で指定された **cn** コンポーネント、**ou** コンポーネント、**o** コンポーネント、および **c** コンポーネントからエントリーの DN を構築し、それを検索 DN として使用してディレクトリー内のエントリーを検索することで、証明書を LDAP ディレクトリーエントリーにマッピングします。マッパーは以下のエントリーを見つけます。

- CA 証明書と CRL を公開するためのディレクトリー内の CA のエントリー。
- エンドエンティティー証明書を公開するためのディレクトリーのエンドエンティティー。

マッパーは DN コンポーネントを取得して検索 DN を構築します。マッパーは任意の root 検索 DN も取得します。サーバーは DN コンポーネントを使用して LDAP エントリーを形成し、サブツリー検索を開始し、フィルターコンポーネントを使用してサブツリーの検索フィルターを形成します。DN コンポーネントが設定されていないと、サーバーはサブツリーにベース DN を使用します。ベース DN が null で、DN コンポーネントが一致しない場合には、エラーが返されます。DN コンポーネントおよびフィルターコンポーネントがいずれも一致しない場合は、エラーが返されます。フィルターコンポーネントが null の場合は、ベース検索が実行されます。

DNComps パラメーターおよび **filterComps** パラメーターは、有効な DN コンポーネントまたは属性をコンマで区切って指定します。パラメーターは、属性の複数のエントリーを受け入れません。たとえば、**filterComps** を **cn,ou** に設定することができますが、**cn,ou2ou1** には設定できません。ディレクトリーエントリーに複数の **ou** が含まれている場合など、同じ属性の複数のインスタンスを持つフィルターを作成するには、**LdapDNCompsMap** モジュールのソースコードを修正します。

以下のコンポーネントは、DN で一般的に使用されます。

- **uid** は、ディレクトリー内のユーザーのユーザー ID を表します。
- **cn** は、ディレクトリー内のユーザーの共通名を表します。
- **ou** は、ディレクトリー内の組織単位を表します。
- **o** は、ディレクトリー内の組織を表します。
- **l** は地域 (都市) を表します。
- **st** は状態を表します。
- **c** は国を表します。

たとえば、次の DN は、米国カリフォルニア州マウンテンビューにある Example Corporation の営業部門で働く Jane Doe というユーザーを表しています。

```
cn=Jane Doe, ou=Sales, o=Example Corporation, l=Mountain View, st=California, c=US
```

証明書マネージャーは、コンポーネント (**cn**、**ou**、**o**、**l**、**st**、および **c**) の一部またはすべてを使用して、ディレクトリーを検索するための DN を構築します。マッパールールを作成するときに、これらのコンポーネントをサーバーが DN の構築に使用するように指定できます。つまり、ディレクトリー内の属性に一致するコンポーネントです。これは **dnComps** パラメーターで設定されます。

たとえば、コンポーネント **cn**、**ou**、**o**、および **c** は、**dnComps** パラメーターの値として 設定されます。ディレクトリー内の Jane Doe のエントリーを見つけるために、Certificate Manager は、証明書から DN 属性値を読み取ることによって次の DN を構築し、ディレクトリーを検索するためのベースとして DN を使用します。

```
cn=Jane Doe, ou=Sales, o=Example Corporation, c=US
```

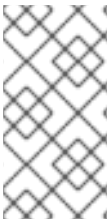
- サブジェクト名には、**dnComps** パラメーターで指定されたすべてのコンポーネントを含める必要はありません。この例の **l** および **st** など、サーバーは、サブジェクト名の一部ではないコンポーネントを無視します。
- 未指定のコンポーネントは、DN の構築には使用されません。例では、**ou** コンポーネントが含まれていない場合、サーバーはこの DN をディレクトリー検索のベースとして使用します。

  ```
  cn=Jane Doe, o=Example Corporation, c=US
  ```

dnComps パラメーターの場合、Certificate Manager が LDAP DN を正確に形成するために使用できる DN コンポーネントを入力します。ただし、特定の状況では、証明書のサブジェクト名がディレクトリー内の複数のエントリーと一致する場合があります。次に、Certificate Manager は、DN から一致するエントリーを 1 つ取得できない可能性があります。たとえば、サブジェクト名 **cn=Jane Doe, ou=Sales, o=Example Corporation, c=US** は、ディレクトリー内の Jane Doe という名前の 2 つのユーザーと一致する可能性があります。その場合、Certificate Manager には、証明書のサブジェクトに対応するエントリーを決定するための追加の基準が必要です。

Certificate Manager がディレクトリー内の異なるエントリーを区別するために使用する必要のあるコンポーネントを指定するには、**filterComps** パラメーターを使用します。詳細は、表 C.10「LdapDNCompsMap 設定パラメーター」 を参照してください。たとえば、**cn**、**ou**、**o**、および **c** が**dnComps** パラメーターの値である場合、**l** 属性を使用して同じ **cn**、**ou**、**o**、および **c** 値を持つエントリーを区別できる場合にのみ、**filterComps** パラメーター に **l** を入力します。

2 つの Jane Doe エントリーが **uid** 属性の値によって区別される場合、(1 つのエントリー**uid** は **janedoe1** で、(もう 1 つのエントリーの **uid** は **janedoe2**)、証明書のサブジェクト名を設定して **uid** コンポーネントを含めるようにできます。



注記

e、**l**、および **st** コンポーネントは、終了エンティティー向けに提供される証明書要求の標準フォームに含まれません。これらのコンポーネントをフォームに追加することも、証明書発行フォームのサブジェクト名を編集するときに発行エージェントにこれらのコンポーネントの挿入を要求することもできます。

C.2.5.1. LdapDNCompsMap の設定パラメーター

この設定では、Certificate Manager は、DN を形成するための **dnComps** 値と、サブツリーの検索フィルターを形成するための **filterComps** 値を使用して、L その証明書と、DAP ディレクトリー内の証明書をマップします。

- フォーム化された DN が null の場合、サーバーはサブツリーの **baseDN** の値を使用します。正式な DN とベース DN の両方が null の場合、サーバーはエラーをログに記録します。
- フィルターが null の場合、サーバーは検索に **baseDN** の値を使用します。フィルターとベース DN の両方が null の場合、サーバーはエラーをログに記録します。

表C.10「LdapDNCompsMap 設定パラメーター」 これらのパラメーターを説明します。

表C.10 LdapDNCompsMap 設定パラメーター

パラメーター	詳細
baseDN	公開ディレクトリー内のエントリーの検索を開始する DN を指定します。**dnComps** フィールドが空の場合、サーバーはベース DN 値を使用してディレクトリーの検索を開始します。
dnComps	公開ディレクトリーのどこで、Certificate Manager が CA またはエンドエンティティーの情報と一致する LDAP エントリーの検索を開始するかを指定します。 たとえば、**dnComps** が、DN の **o** 属性および **c** 属性を使用する場合、サーバーは、ディレクトリーで **o=org**、**c=country** エントリーから検索を開始し、**org** と **country** を証明書の DN の値に置き換えます。 **dnComps** フィールドが空の場合、サーバーは、**baseDN** フィールドを確認し、**filterComps** パラメーター値により指定されるフィルターに一致するエントリーの DN によって指定されるディレクトリーツリーを検索します。 許容値は、有効な DN コンポーネントまたは属性をコンマで区切って指定します。

パラメーター	詳細
filterComps	Certificate Manager が検索結果からエントリーをフィルタリングするために使用するコンポーネントを指定します。サーバーは **filterComps** 値を使用して、サブツリーの LDAP 検索フィルターを形成します。サーバーは、証明書のサブジェクト名からこれらの属性の値を収集することにより、フィルターを構築します。フィルターを使用して、LDAP ディレクトリー内のエントリーを検索して照合します。 サーバーが証明書から収集した情報と一致するエントリーをディレクトリー内で複数検出した場合、検索は成功し、サーバーはオプションで検証を実行します。たとえば、**filterComps** が電子メールおよびユーザー ID 属性を使用するように設定されていると (**filterComps=e,uid**)、サーバーはディレクトリーを検索して、電子メールとユーザー ID の値が証明書から収集された情報と一致するエントリーを検索します。 許容値は、コンマで区切られた証明書 DN 内の有効なディレクトリー属性です。フィルターの属性名は、LDAP ディレクトリー内の属性名ではなく、証明書の属性名である必要があります。たとえば、ほとんどの証明書には、ユーザーの電子メールアドレスの **e** 属性があります。LDAP は、その属性の **mail** を呼び出します。

C.3. ルールインスタンス

このセクションでは、設定したルールインスタンスを説明します。

C.3.1. LdapCaCertRule

LdapCaCertRule は、CA 証明書を LDAP ディレクトリーに公開するために使用することができます。

表C.11 LdapCaCert Rule 設定パラメーター

パラメーター	値	詳細
type	**cacert**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCaCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCaCertMap」を参照してください。

パラメーター	値	詳細
publisher	**LdapCaCertPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapCaCertPublisher」 を参照してください。

C.3.2. LdapXCertRule

LdapXCertRule は、LDAP ディレクトリーにコピー間の証明書を公開するのに使用されます。

表C.12 LdapXCert Rule 設定パラメーター

パラメーター	値	詳細
type	**xcert**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCaCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCaCertMap」を参照してください。
publisher	**LdapCrossCertPairPublisher**	ルールで使用するパブリッシャーを指定します。このパブリッシャーのの詳細は、「LdapCertificatePairPublisher」 を参照してください。

C.3.3. LdapUserCertRule

LdapUserCertRule は、ユーザー証明書を LDAP ディレクトリーに公開するために使用されます。

表C.13 LdapUserCert Rule 設定パラメーター

パラメーター	値	詳細
type	**certs**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。

パラメーター	値	詳細
enable	**はい**	ルールを有効にします。
mapper	**LdapUserCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapSimpleMap」を参照してください。
publisher	**LdapUserCertPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapUserCertPublisher」を参照してください。

C.3.4. LdapCRLRule

LdapCRLRule は CRL を LDAP ディレクトリーに公開するために使用されます。

表C.14 LdapCRL ルール設定パラメーター

パラメーター	値	詳細
type	**crl**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCrlMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCrlMap」を参照してください。
publisher	**LdapCrlPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapCrlPublisher」を参照してください。

付録D ACL リファレンス

このセクションでは、各リソースが制御するものについて説明し、それらの操作の結果を説明する可能な操作をリストし、定義された各 ACL リソースのデフォルトの ACI を提供します。各サブシステムには、そのサブシステムに関連する ACL のみが含まれます。

D.1. ACL 設定ファイルについて

アクセス制御 は、サーバーの一部にアクセスできるユーザーと、ユーザーが実行できる操作に関するルールを設定する方法です。LDAP ディレクトリーサービスに依存し、Java コンソールを使用する 4 つのサブシステム (CA、KRA、OCSP、および TKS) はすべて、LDAP スタイルのアクセス制御を実装してリソースにアクセスします。これらの **アクセス制御リスト** (ACL)は、インスタンスの **/var/lib/pki/*instance_name*/conf** ディレクトリーの **acl.ldif** ファイルにあります。

注記

本セクションでは、アクセス制御の概念の概要を説明します。アクセス制御については、『Red Hat Directory Server Administration Guide 』 の 『Managing Access Control』 で詳しく説明されています。

Certificate System ACL ファイルは、内部データベースにより読み込まれる LDIF ファイルです。個々の ACL は、***resourceACLS*** 属性として定義されます。これは、保護されているサブシステムの領域を識別する属性と、その後設定されているすべての特定のアクセス制御のリストで識別されます。

```
resourceACLS: class_name:all rights: allow|deny (rights)  type=target description
```

リソースへのアクセスを許可または拒否する各ルールは、**アクセス制御命令** (ACI) と呼ばれます。(リソースの ACI の合計はアクセス制御リストです。) 実際の ACI を定義する前に、ACL 属性は、Certificate System サブシステムによって使用される特定のプラグインクラスに最初に適用されます。これにより、各 ACL がサブシステムによって実行される特定の機能に集中し、インスタンスのセキュリティーが強化され、ACL の適用をより適切に制御できるようになります。

例D.1 証明書プロファイルの一覧を表示するデフォルト ACL

```
resourceACLS: certServer.ca.profiles:list:allow (list)  group="Certificate Manager
Agents":Certificate Manager agents may list profiles
```

各サブシステム (CA、KRA、OCSP、および TKS) には操作用の異なるリソースがあるため、各サブシステムインスタンスには独自の **acl.ldif** ファイルと独自に定義された ACL があります。

各 ACI は、実行できるアクセスまたは動作 (**右**)、と ACI の適用対象 (**ターゲット**) を定義します。ACI の基本的な形式は次のとおりです。

```
allow|deny (rights) user|group
```

権限 は、ACI がユーザーに実行を許可する操作のタイプです。LDAP ACI の場合は、検索、読み取り、書き込み、削除など、ディレクトリーエントリーに対する権限のリストは比較的限られています。Certificate System は、取り消し、送信、割り当てなどの一般的な PKI タスクを扱う追加の権限を使用します。

操作が ACI で明示的に許可されていない場合、この操作は暗黙的に拒否されます。1 つの ACI で操作が明示的に拒否された場合は、明示的に許可されている ACI よりも優先されます。拒否ルールは、ルールが追加のセキュリティーを提供できるようにするために常に優れています。

各 ACI は特定のユーザーまたはグループに適用する必要があります。エントリーベースのアクセスではなくクライアントベースのアクセスを定義する **ipaddress=** などのオプションがありますが、これは、いくつかの一般的な条件 (通常 **user=** または **group=**) を使用して設定されます。複数の条件がある場合は、論理和 ("or") を表す二重パイプ (||) 演算子と、論理積 ("and") を表す二重アンパサンド (&&) 演算子を使用して条件を設定することができます。たとえば、**group="group1" || "group2"** となります。

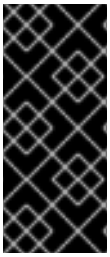
resourceACLS 属性値の各領域は、表D.1「ACL 属性値のセクション」 で定義されます。

表D.1 ACL 属性値のセクション

値	説明
class_name	ACI が適用されるプラグインクラス。
すべての操作	ACI 定義に含まれるすべての操作の一覧。1 つの ACI には複数の操作があり、1 つの ***resourceACLS*** 属性に複数の ACI があります。
allow|deny	アクションがターゲットユーザーまたはグループに対して許可されているか、ターゲットユーザーまたはグループに対して拒否されているか。
(operations)	許可または拒否されている操作。
type=target	これが適用されるユーザーを特定するターゲット。これは一般的にユーザー (**user=**"name" など) またはグループ (**group=**"group" など) です。条件が複数ある場合は、二重パイプ (||) 演算子 (論理 "or") と二重アンパサンド (&&) 演算子 (論理 "and") を使用して条件を設定することができます。たとえば、**group="group1" || "group2"** となります。
説明	ACL が実行していることの説明。

D.2. 共通 ACL

このセクションでは、4 つのサブシステムタイプすべてに共通するデフォルトのアクセス制御設定を説明します。これらのアクセス制御ルールは、ユーザーやグループのログ記録や追加など、基本的で一般的な設定設定へのアクセスを管理します。



重要

これらの ACL は、各サブシステムインスタンスの **acl.ldif** ファイルで同じ ACL が発生するのが一般的です。これらは、設定ファイルまたは設定がすべてのサブシステムインスタンスによって共通に保持されるという意味で、**共有** ACL ではありません。他のすべてのインスタンス設定と同様に、これらの ACL は、インスタンス固有の **acl.ldif** ファイルで、他のサブシステムインスタンスとは独立して維持されます。

D.2.1. certServer.acl.configuration

ACL 設定への操作を制御します。デフォルト設定は以下のようになります。

```
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"
```

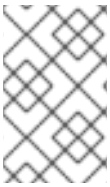
表D.2 certServer.acl.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ACL リソースを表示し、ACL リソース、ACL リストエバリュエーター、および ACL エバリュエータータイプを一覧表示します。	許可	管理者 エージェント 監査者
modify	ACL エバリュエーターの追加、削除、および更新。	許可	管理者

D.2.2. certServer.admin.certificate

Certificate Manager から証明書をインポートするユーザーを制御します。デフォルトでは、この操作はすべてのユーザーが許可されます。デフォルト設定は以下のようになります。

```
allow (import) user="anybody"
```



注記

このエントリーは、インスタンスの設定に使用される CA 管理 Web インターフェイスに関連付けられています。この ACL は、インスタンスの設定中にのみ使用可能であり、CA の実行後には使用できません。

表D.3 certServer.admin.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
import	CA 管理者証明書をインポートして、シリアル番号で証明書を取得します。	許可	全ユーザー

D.2.3. certServer.auth.configuration

認証設定の操作を制御します。

```
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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	認証プラグイン、認証タイプ、設定済みの認証マネージャープラグイン、および認証インスタンスを表示します。認証マネージャープラグインおよび認証マネージャーインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	認証プラグインおよび認証インスタンスを追加または削除します。認証インスタンスを変更します。	許可	管理者

D.2.4. certServer.clone.configuration

クローン作成で使用される設定情報を読み取り、変更できるユーザーを制御します。デフォルト設定は次のとおりです。

```
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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	元のインスタンス設定を表示します。	許可	エンタープライズ管理者
modify	元のインスタンス設定を変更します。	許可	エンタープライズ管理者

D.2.5. certServer.general.configuration

CA の設定を表示および編集できるユーザーなど、サブシステムインスタンスの一般的な設定へのアクセスを制御します。

> 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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	管理用の運用環境、LDAP 設定、SMTP 設定、サーバー統計、暗号化、トークン名、証明書のサブジェクト名、証明書のニックネーム、サーバーによって読み込むすべてのサブシステム、CA 証明書、およびすべての証明書を表示します。	許可	管理者 エージェント 監査者
modify	LDAP データベース、SMTP、および暗号化の設定を変更します。インポート証明書の発行、証明書のインストール、CA 証明書の信頼と信頼解除、クロスペア証明書のインポート、および証明書の削除。サーバーの再起動および停止操作を実行します。すべてのトークンにログインして、トークンのステータスを確認します。オンデマンドでセルフテストを実行します。証明書情報を取得します。証明書サブジェクト名を処理します。証明書サブジェクト名、証明書キーの長さ、および証明書拡張機能を検証します。	許可	管理者

D.2.6. certServer.log.configuration

ログ設定の変更など、Certificate Manager のログ設定へのアクセスを制御します。

> 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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログプラグインの情報、ログプラグイン設定、およびログインスタンス設定を表示します。ログプラグインおよびログインスタンスを一覧表示します (NTpixel を除く)。	許可	管理者 エージェント 監査者
modify	ログプラグインおよびログインスタンスを追加し、削除します。ログローロールオーバーパラメーターやログレベルなど、ログインスタンスを変更します。	許可	管理者

D.2.7. certServer.log.configuration.fileName

インスタンスのログのファイル名を変更するアクセスを制限します。

```
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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログインスタンスの **fileName** パラメーターの値を表示します。	許可	管理者 エージェント 監査者
modify	ログインスタンスの **fileName** パラメーターの値を表示します。	却下	全ユーザー

D.2.8. certServer.log.content.system

インスタンスのログを表示できるユーザーを制御します。

```
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 の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログの内容を表示します。すべてのログを一覧表示します。	許可	管理者 エージェント 監査者

D.2.9. certServer.log.content.transactions

インスタンスのトランザクションログを表示できるユーザーを制御します。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors"
```

表D.10 certServer.log.content.transactions ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログの内容を表示します。すべてのログを一覧表示します。	許可	管理者 エージェント 監査者

D.2.10. certServer.log.content.signedAudit

署名付き監査ログにアクセスできるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (read) group="Auditors"
```

表D.11 certServer.log.content.signedAudit ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログの内容を表示します。ログを一覧表示します。	許可	監査者

D.2.11. certServer.registry.configuration

プラグインモジュールの登録に使用されるファイルである管理レジストリーへのアクセスを制御します。現在、これは証明書プロファイルプラグインの登録にのみ使用されます。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.12 certServer.registry.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	管理レジストリー、サポートされているポリシー制約、プロファイルプラグインの設定、およびプロファイルプラグインのリストを表示します。	許可	管理者 エージェント 監査者
modify	個々のプロファイル実装プラグインを登録します。	許可	管理者

D.3. 証明書マネージャー固有の ACL

本セクションでは、Certificate Manager 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。CA ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれています。

CA の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、証明書の一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.3.1. certServer.admin.ocsp

Certificate Manager の OCSP 設定へのアクセスを、エンタープライズ OCSP 管理者グループのメンバーに制限します。

```
allow (modify,read) group="Enterprise OCSP Administrators"
```

表D.13 certServer.admin.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	OCSP 設定、OCSP ストア設定、およびデフォルトの OCSP ストアを変更します。	許可	エンタープライズ OCSP 管理者

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP 設定を読み取ります。	許可	エンタープライズ OCSP 管理者

D.3.2. certServer.ca.certificate

証明書のインポートや取り消しなど、エージェントサービスインターフェイスでの証明書の基本的な管理操作を制御します。デフォルト設定は以下のようになります。

```
allow (import,unrevoke,revoke,read) group="Certificate Manager Agents"
```

表D.14 certServer.ca.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
import	シリアル番号で証明書を取得します。	許可	Certificate Manager Agent
unrevoke	証明書のステータスを失効から変更します。	許可	Certificate Manager Agent
revoke	証明書のステータスを失効に変更します。	許可	Certificate Manager Agent
read	リクエスト ID に基づいて証明書を取得し、リクエスト ID またはシリアル番号に基づいて証明書の詳細を表示します。	許可	Certificate Manager Agent

D.3.3. certServer.ca.certificates

エージェントサービスインターフェイスを介して証明書の一覧表示または取り消しを行う操作を制御します。デフォルト設定は以下のようになります。

```
allow (revoke,list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

表D.15 certServer.ca.certificates ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
revoke	証明書を取り消すか、または証明書失効リスト要求を承認します。TPS から証明書を取り消します。失効要求に関する追加データのプロンプトを表示します。	許可	Certificate Manager Agent 登録マネージャーエージェント
list	検索に基づいて証明書を一覧表示します。シリアル番号の範囲に基づいて証明書の範囲の詳細を取得します。	許可	Certificate Manager Agent 登録マネージャーエージェント

D.3.4. certServer.ca.configuration

Certificate Manager の一般的な設定での操作を制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.16 certServer.ca.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	CRL プラグイン情報、一般的な CA 設定、CA コネクター設定、CRL 発行ポイント設定、CRL プロファイル設定、要求通知設定、失効通知設定、キュー内要求通知設定、および CRL 拡張設定を表示します。CRL 拡張設定および CRL 発行ポイント設定を一覧表示します。	許可	管理者 エージェント 監査者

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	CRL 発行ポイントを追加し、削除します。一般的な CA 設定、CA コネクター設定、CRL 発行ポイント設定、CRL 設定、要求通知設定、失効通知設定、キュー内要求通知設定、および CRL 拡張設定を変更します。	許可	管理者

D.3.5. certServer.ca.connector

特別なコネクターを CA に送信する操作を制御します。デフォルト設定は以下のようになります。

```
allow (submit) group="Trusted Managers"
```

表D.17 certServer.ca.connector ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	リモート信頼できるマネージャーからのリクエストを送信します。	許可	信頼できるマネージャー

D.3.6. certServer.ca.connectorInfo

コネクター情報へのアクセスを制御して、CA と KRA と間の信頼できる関係を管理します。これらの信頼関係は、CA と KRA が自動的に接続して、主要なアーカイブおよび復元操作を実行できるようにする特別な設定です。これらの信頼関係は、特別なコネクタープラグインを使用して設定されます。

```
allow (read) group="Enterprise KRA Administrators";allow (modify) group="Enterprise KRA Administrators" || group="Subsystem Group"
```

表D.18 certServer.ca.connectorInfo ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	コネクタープラグイン設定を読み取ります。	許可	エンタープライズ KRA 管理者

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	コネクタープラグイン設定を変更します。	許可	エンタープライズ KRA 管理者 サブシステムグループ

D.3.7. certServer.ca.crl

エージェントサービスインターフェイスを介して CRL の読み取りまたは更新へのアクセスを制御します。デフォルト設定は次のとおりです。

```
allow (read,update) group="Certificate Manager Agents"
```

表D.19 certServer.ca.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	CRL を表示し、CA CRL 処理に関する詳細情報を取得します。	許可	Certificate Manager Agent
update	CRL を更新します。	許可	Certificate Manager Agent

D.3.8. certServer.ca.directory

証明書および CRL の公開に使用される LDAP ディレクトリーへのアクセスを制御します。

```
allow (update) group="Certificate Manager Agents"
```

表D.20 certServer.ca.directory ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
update	CA 証明書、CRL、およびユーザー証明書を LDAP ディレクトリーに公開します。	許可	Certificate Manager Agent

D.3.9. certServer.ca.group

Certificate Manager インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.21 certServer.ca.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。属性内でユーザー証明書の追加または変更	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.3.10. certServer.ca.ocsp

エージェントサービスインターフェイスを介して、使用統計などの OCSP 情報にアクセスして読み取る機能を制御します。

```
allow (read) group="Certificate Manager Agents"
```

表D.22 certServer.ca.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP 使用状況の統計を取得します。	許可	Certificate Manager Agent

D.3.11. certServer.ca.profile

エージェントサービスページで証明書プロファイル設定へのアクセスを制御します。

```
allow (read,approve) group="Certificate Manager Agents"
```

表D.23 certServer.ca.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書プロファイルの詳細を表示します。	許可	Certificate Manager Agent
approve	証明書プロファイルを承認し、有効にします。	許可	Certificate Manager Agent

D.3.12. certServer.ca.profiles

エージェントサービスインターフェイスで証明書プロファイルを一覧表示するアクセスを制御します。

```
allow (list) group="Certificate Manager Agents"
```

表D.24 certServer.ca.profiles ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	証明書プロファイルの一覧表示。	許可	Certificate Manager Agent

D.3.13. certServer.ca.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.25 certServer.ca.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新しいエージェントを登録します。	許可	エンタープライズ管理者
read	既存のエージェント情報を読み取ります。	許可	エンタープライズ管理者

D.3.14. certServer.ca.request.enrollment

登録リクエストの処理方法および割り当て方法を制御します。デフォルト設定は次のとおりです。

```
allow (submit) user="anybody";allow (read,execute,assign,unassign) group="Certificate Manager
Agents"
```

表D.26 certServer.ca.request.enrollment ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	登録リクエストを表示します。	許可	Certificate Manager Agent

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
execute	リクエストの承認状態を変更します。	許可	Certificate Manager Agent
submit	リクエストを送信します。	許可	Anybody
assign	Certificate Manager エージェントに要求を割り当てます。	許可	Certificate Manager Agent
unassign	要求の割り当てを変更します。	許可	Certificate Manager Agent

D.3.15. certServer.ca.request.profile

証明書プロファイルベースの要求の処理を制御します。デフォルト設定は次のとおりです。

```
allow (approve,read) group="Certificate Manager Agents"
```

表D.27 certServer.ca.request.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
approve	証明書プロファイルベースの証明書要求の承認状態を変更します。	許可	Certificate Manager Agent
read	証明書プロファイルベースの証明書要求を表示します。	許可	Certificate Manager Agent

D.3.16. certServer.ca.requests

エージェントサービスインターフェイスで証明書要求を一覧表示できるユーザーを制御します。

```
allow (list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

表D.28 certServer.ca.requests ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	要求の範囲の詳細を取得し、複雑なフィルターを使用して証明書を検索します。	許可	Certificate Manager Agent 登録マネージャーエージェント

D.3.17. certServer.ca.systemstatus

Certificate Manager インスタンスの統計を表示できるユーザーを制御します。

```
allow (read) group="Certificate Manager Agents"
```

表D.29 certServer.ca.systemstatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	統計を表示します。	許可	Certificate Manager Agent

D.3.18. certServer.ee.certchain

エンドエンティティーの CA 証明書チェーンにアクセスできるユーザーを制御します。

```
allow (download,read) user="anybody"
```

表D.30 certServer.ee.certchain ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
download	CA の証明書チェーンをダウンロードします。	許可	全ユーザー
read	CA の証明書チェーンを表示します。	許可	全ユーザー

D.3.19. certServer.ee.certificate

エンドエンティティーページを介して、証明書のインポートや取り消しなどのほとんどの操作で、証明書にアクセスできるユーザーを制御します。

```
allow (renew,revoke,read,import) user="anybody"
```

表D.31 certServer.ee.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
renew	既存の証明書を更新する要求を送信します。	許可	全ユーザー
revoke	ユーザー証明書の失効要求を送信します。	許可	全ユーザー
read	証明書のシリアル番号または要求 ID に基づいて証明書を取得して表示します。	許可	全ユーザー
import	シリアル番号に基づいて証明書をインポートします。	許可	全ユーザー

D.3.20. certServer.ee.certificates

失効した証明書を一覧表示したり、エンドエンティティーに失効リクエストを送信できるユーザーを制御します。

```
allow (revoke,list) user="anybody"
```

表D.32 certServer.ee.certificates ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
revoke	取り消しする証明書の一覧を送信します。	許可	取り消される証明書の対象は、CA に認証するために提示された証明書と一致させる必要があります。
list	指定の基準に一致する証明書を検索します。	許可	全ユーザー

D.3.21. certServer.ee.crl

エンドエンティティーページから CRL へのアクセスを制御します。

```
allow (read,add) user="anybody"
```

表D.33 certServer.ee.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書失効リストを取得および表示します。	許可	全ユーザー
add	CRL を OCSP サーバーに追加します。	許可	全ユーザー

D.3.22. certServer.ee.profile

プロファイルの詳細を表示したり、プロファイルを介してリクエストを送信したりできるユーザーなど、エンドエンティティーページの証明書プロファイルへのアクセスを制御します。

```
allow (submit,read) user="anybody"
```

表D.34 certServer.ee.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	証明書プロファイルを使用して証明書要求を送信します。	許可	全ユーザー
read	証明書プロファイルの詳細の表示	許可	全ユーザー

D.3.23. certServer.ee.profiles

エンドエンティティーページでアクティブな証明書プロファイルを一覧表示できるユーザーを制御します。

```
allow (list) user="anybody"
```

表D.35 certServer.ee.profiles ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	証明書プロファイルの一覧表示。	許可	全ユーザー

D.3.24. certServer.ee.request.ocsp

クライアントが OCSP 要求を送信する IP アドレスに基づいてアクセスを制御します。

```
allow (submit) ipaddress=".*"
```

表D.36 certServer.ee.request.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	OCSP 要求を送信します。	許可	すべての IP アドレス

D.3.25. certServer.ee.request.revocation

エンドエンティティーページで証明書失効リスト要求を送信できるユーザーを制御します。

```
allow (submit) user="anybody"
```

表D.37 certServer.ee.request.revocation ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	証明書を取り消す要求を送信します。	許可	全ユーザー

D.3.26. certServer.ee.requestStatus

エンドエンティティーページで証明書要求のステータスを表示できるユーザーを制御します。

```
allow (read) user="anybody"
```

表D.38 certServer.ee.requestStatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	その要求に対して発行された証明書の要求およびシリアル番号を取得します。	許可	全ユーザー

D.3.27. certServer.job.configuration

Certificate Manager にジョブを設定できるユーザーを制御します。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.39 certServer.job.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	基本的なジョブ設定、ジョブインスタンス設定、ジョブプラグイン設定を表示します。ジョブプラグインおよびジョブインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	ジョブプラグインおよびジョブインスタンスを追加および削除します。ジョブプラグインとジョブインスタンスを変更します。	許可	管理者

D.3.28. certServer.profile.configuration

証明書プロファイル設定へのアクセスを制御します。デフォルト設定は次のとおりです。

> allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"

表D.40 certServer.profile.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書プロファイルのデフォルトと制約、入力、出力、入力設定、出力設定、デフォルト設定、ポリシー制約設定、および証明書プロファイルインスタンス設定を表示します。証明書プロファイルプラグインおよび証明書プロファイルインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	証明書プロファイルのデフォルトおよび制約、入力、出力、および証明書プロファイルインスタンスの追加、変更、削除を行います。デフォルトのポリシー制約設定を追加および変更します。	許可	管理者

D.3.29. certServer.publisher.configuration

Certificate Manager の公開設定を表示および編集できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";allow (modify) group="Administrators"
```

表D.41 certServer.publisher.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	LDAP サーバーの宛先情報、パブリッシャープラグイン設定、パブリッシャーインスタンス設定、マッパープラグイン設定、マッパーインスタンス設定、ルールプラグイン設定、およびルールインスタンス設定を表示します。パブリッシャープラグインとインスタンス、ルールプラグインとインスタンス、およびマッパープラグインとインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	パブリッシャープラグイン、パブリッシャーインスタンス、マッパープラグイン、マッパーインスタンス、ルールプラグイン、およびルールインスタンスを追加および削除します。パブリッシャーインスタンス、マッパーインスタンス、ルールインスタンス、および LDAP サーバーの宛先情報を変更します。	許可	管理者

D.3.30. certServer.securitydomain.domainxml

ドメインホストの Certificate Manager によってレジストリーに保持されるセキュリティードメイン情報へのアクセスを制御します。セキュリティードメイン設定は、設定中にサブシステムインスタンスによって直接アクセスおよび変更されるため、サブシステムへの適切なアクセスを常に許可する必要があります。そうしないと、設定が失敗する可能性があります。

```
allow (read) user="anybody";allow (modify) group="Subsystem Group"
```

表D.42 certServer.securitydomain.domainxml ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	セキュリティードメイン設定を表示します。	許可	Anybody
modify	インスタンス情報を変更し、インスタンスを追加および削除して、セキュリティードメイン設定を変更します。	許可	サブシステムグループ エンタープライズ管理者

D.4. キーリカバリー認証局固有の ACL

本セクションでは、KRA 向けに特別に適用されるデフォルトのアクセス制御設定を説明します。KRA ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれています。

KRA の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、キーの一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.4.1. certServer.job.configuration

KRA のジョブを設定できるユーザーを制御します。

```
allow (read) group="Administrators" || group="Key Recovery Authority Agents" ||
group="Auditors";allow (modify) group="Administrators"
```

表D.43 certServer.job.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	基本的なジョブ設定、ジョブインスタンス設定、ジョブプラグイン設定を表示します。ジョブプラグインおよびジョブインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	ジョブプラグインおよびジョブインスタンスを追加および削除します。ジョブプラグインとジョブインスタンスを変更します。	許可	管理者

D.4.2. certServer.kra.certificate.transport

KRA のトランスポート証明書を表示できるユーザーを制御します。

```
allow (read) user="anybody"
```

表D.44 certServer.kra.certificate.transport ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	KRA インスタンスのトランスポート証明書を表示します。	許可	全ユーザー

D.4.3. certServer.kra.configuration

KRA の設定を設定および管理するユーザーを制御します。

```
allow (read) group="Administrators" || group="Auditors" || group="Key Recovery Authority Agents" ||
allow (modify) group="Administrators"
```

表D.45 certServer.kra.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	必要なリカバリーエージェントの承認の数を確認します。	許可	管理者 エージェント 監査者
modify	必要なリカバリーエージェントの承認の数を変更します。	許可	管理者

D.4.4. certServer.kra.connector

KRA に接続するために CA に設定された特別なコネクターで要求を送信できるエンティティーを制御します。デフォルト設定は以下のようになります。

```
allow (submit) group="Trusted Managers"
```

表D.46 certServer.kra.connector ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	新しい鍵のアーカイブ要求を送信します (TMS 以外)。	許可	信頼できるマネージャー

D.4.5. certServer.kra.GenerateKeyPair

KRA にキーリカバリー要求を送信できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (execute) group="Key Recovery Authority Agents"
```

表D.47 certServer.kra.GenerateKeyPair ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	サーバー側のキー生成 (TMS のみ) を実行します。	許可	KRA エージェント

D.4.6. certServer.kra.getTransportCert

KRA にキーリカバリー要求を送信できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (download) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.48 certServer.kra.getTransportCert ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
download	KRA トランスポート証明書を取得します。	許可	エンタープライズ管理者

D.4.7. certServer.kra.group

KRA インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.49 certServer.kra.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.4.8. certServer.kra.key

鍵の表示、リカバリー、またはダウンロードを使用して鍵情報にアクセスできるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read,recover,download) group="Key Recovery Authority Agents"
```

表D.50 certServer.kra.key ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	重要なアーカイブ記録に関する公開情報を表示します。	許可	KRA エージェント
recover	データベースからキー情報を取得してリカバリー操作を実行します。	許可	KRA エージェント
download	エージェントサービスページからキー情報をダウンロードします。	許可	KRA エージェント

D.4.9. certServer.kra.keys

エージェントサービスページからアーカイブされた鍵を一覧表示できるユーザーを制御します。

```
allow (list) group="Key Recovery Authority Agents"
```

表D.51 certServer.kra.keys ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	アーカイブされた鍵の範囲を検索し、一覧表示します。	許可	KRA エージェント

D.4.10. certServer.kra.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.52 certServer.kra.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新規ユーザーを登録します。	許可	エンタープライズ管理者
read	既存のユーザー情報を読み込みます。	許可	エンタープライズ管理者

D.4.11. certServer.kra.request

エージェントサービスインターフェイスでキーのアーカイブとリカバリー要求を表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.53 certServer.kra.request ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	鍵のアーカイブまたはリカバリー要求を表示します。	許可	KRA エージェント

D.4.12. certServer.kra.request.status

エンドエンティティーページでキーリカバリー要求のステータスを表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.54 certServer.kra.request.status ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	エージェントサービスページでキーリカバリー要求のステータスを取得します。	許可	KRA エージェント

D.4.13. certServer.kra.requests

エージェントサービスインターフェイスでキーのアーカイブとリカバリー要求を一覧表示できるユーザーを制御します。

```
allow (list) group="Key Recovery Authority Agents"
```

表D.55 certServer.kra.requests ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	キーアーカイブおよびリカバリー要求の範囲の詳細を取得します。	許可	KRA エージェント

D.4.14. certServer.kra.systemstatus

KRA インスタンスの統計を表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.56 certServer.kra.systemstatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	統計を表示します。	許可	KRA エージェント

D.4.15. certServer.kra.TokenKeyRecovery

トークンのキーリカバリー要求を KRA に送信するユーザーを制御します。これは、失われたトークンを置き換えるための一般的なリクエストです。デフォルト設定は以下のようになります。

```
allow (submit) group="Key Recovery Authority Agents"
```

表D.57 certServer.kra.TokenKeyRecovery ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	トークンリカバリーのキーリカバリー要求を送信または開始します。	許可	KRA エージェント

D.5. オンライン証明書ステータスマネージャー固有の ACL

本セクションでは、Online Certificate Status Manager 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。OCSP レスポンダーの ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれます。

OCSP の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、CRL の一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.5.1. certServer.ee.crl

エンドエンティティーページから CRL へのアクセスを制御します。

```
allow (read) user="anybody"
```

表D.58 certServer.ee.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書失効リストを取得および表示します。	許可	全ユーザー

D.5.2. certServer.ee.request.ocsp

クライアントが OCSP 要求を送信する IP アドレスに基づいてアクセスを制御します。

```
allow (submit) ipaddress=".*"
```

表D.59 certServer.ee.request.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	OCSP 要求を送信します。	許可	すべての IP アドレス

D.5.3. certServer.ocsp.ca

OCSP レスポンダーを指示できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (add) group="Online Certificate Status Manager Agents"
```

表D.60 certServer.ocsp.ca ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
Add	新しい CA に対する OCSP 要求に応答するように OCSP レスポンダーに指示します。	許可	OCSP Manager エージェント

D.5.4. certServer.ocsp.cas

CRL を Online Certificate Status Manager に公開するすべての Certificate Manager を、エージェントサービスインターフェイスで一覧表示できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (list) group="Online Certificate Status Manager Agents"
```

表D.61 certServer.ocsp.cas ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	OCSP レスポンダーに CRL を公開する Certificate Manager の一覧を表示します。	許可	エージェント

D.5.5. certServer.ocsp.certificate

証明書のステータスを検証できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (validate) group="Online Certificate Status Manager Agents"
```

表D.62 certServer.ocsp.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
validate	指定の証明書の状態を検証します。	許可	OCSP エージェント

D.5.6. certServer.ocsp.configuration

Certificate Manager の OCSP サービスの設定へのアクセス、表示、または変更が可能なユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Online Certificate Status Manager Agents" ||
group="Auditors";allow (modify) group="Administrators"
```

表D.63 certServer.ocsp.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP プラグインの情報、OCSP 設定、および OCSP ストア設定を表示します。OCSP ストアの設定を一覧表示します。	許可	管理者 オンライン証明書ステータスマネージャーエージェント 監査者
modify	OCSP 設定、OCSP ストア設定、およびデフォルトの OCSP ストアを変更します。	許可	管理者

D.5.7. certServer.ocsp.crl

エージェントサービスインターフェイスを介して CRL の読み取りまたは更新へのアクセスを制御します。デフォルト設定は次のとおりです。

```
allow (add) group="Online Certificate Status Manager Agents" || group="Trusted Managers"
```

表D.64 certServer.ocsp.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
add	新しい CRL を、OCSP レスポンダーが管理するものに追加します。	許可	OCSP エージェント 信頼できるマネージャー

D.5.8. certServer.ocsp.group

Online Certificate Status Manager インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.65 certServer.ocsp.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.5.9. certServer.ocsp.info

OCSP レスポンダーに関する情報を読み取ることができるユーザーを制御します。

```
allow (read) group="Online Certificate Status Manager Agents"
```

表D.66 certServer.ocsp.info ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP レスポンダー情報を表示します。	許可	OCSP エージェント

D.6. トークンキーサービス固有の ACL

本セクションでは、トークンキーサービス (TKS) 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。TKS ACL 設定には、「共通 ACL」 にリストされている共通 ACL がすべて含まれます。

TKS の管理コンソール、およびその他のサブシステムによる TKS へのアクセスに対するアクセス制御ルールを設定できます。

D.6.1. certServer.tks.encrypteddata

データを暗号化できるユーザーを制御します。

```
allow(execute) group="Token Key Service Manager Agents"
```

表D.67 certServer.tks.encrypteddata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	TKS に保存されている暗号化されたデータ。	許可	TKS エージェント

D.6.2. certServer.tks.group

TKS インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.68 certServer.tks.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.6.3. certServer.tks.importTransportCert

TKS が鍵の送信に使用するトランスポート証明書をインポートするユーザーを制御します。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.69 certServer.tks.importTransportCert ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	トランスポート証明書を更新します。	許可	エンタープライズ管理者
read	トランスポート証明書をインポートします。	許可	エンタープライズ管理者

D.6.4. certServer.tks.keysetdata

TKS が派生して保存する鍵セットに関する情報を表示するユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.70 certServer.tks.keysetdata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	コンマ区切りのキーセットデータを作成します。	許可	TKS エージェント

D.6.5. certServer.tks.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.71 certServer.tks.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新しいエージェントを登録します。	許可	エンタープライズ管理者
read	既存のエージェント情報を読み取ります。	許可	エンタープライズ管理者

D.6.6. certServer.tks.sessionkey

TPS に接続するために TKS インスタンスが使用するセッションキーを作成するユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.72 certServer.tks.sessionkey ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	TKS が生成するセッションキーを作成します。	許可	TKS エージェント

D.6.7. certServer.tks.randomdata

ランダムなデータを作成できるユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.73 certServer.tks.randomdata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	ランダムなデータを生成します。	許可	TKS エージェント

D.7. TPS 固有の ACL

このセクションでは、トークン処理システム (TPS) 用に特別に設定されたデフォルトのアクセス制御設定属性について説明します。TPS ACL 設定には、「共通 ACL」 にリストされている一般的な ACL もすべて含まれています。

D.7.1. certServer.tps.account

ユーザーがログインおよびログアウトできることを制御します。

```
allow (login,logout) user="anybody"
```

表D.74 certServer.tps.account ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
login	TPS にログインする	許可	全てのユーザー
logout	TPS からログアウトします。	許可	全てのユーザー

D.7.2. certServer.tps.authenticators

管理者のみがオーセンティケーターにアクセスできることを制御します。

```
allow (read,change-status,add,modify,remove) group="Administrators"
```

表D.75 certServer.tps.authenticators ACL 概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	読み取りオーセンティケーター	許可	管理者
change-status	オーセンティケーターのステータスを変更する	許可	管理者
add	オーセンティケーターの追加	許可	管理者

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	オーセンティケーターの更新	許可	管理者
remove	オーセンティケーターの削除	許可	管理者

D.7.3. certServer.tps.audit

管理者のみが監査設定にアクセスできるように制御します。

```
allow (read,modify) group="Administrators"
```

表D.76 certServer.tps.audit ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	設定監査設定の読み取り	許可	管理者
modify	設定監査設定の更新	許可	管理者

D.7.4. certServer.tps.config

管理者のみが設定にアクセスできるように制御します。

```
allow (read,modify) group="Administrators"
```

表D.77 certServer.tps.config ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	設定設定の読み取り	許可	管理者
modify	設定設定の更新	許可	管理者

D.7.5. certServer.tps.connectors

管理者のみがコネクターにアクセスできるように制御します。

```
allow (read,change-status,add,modify,remove) group="Administrators"
```

表D.78 certServer.tps.connectors ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	コネクターの読み取り	許可	管理者
change-status	コネクターのステータスを変更する	許可	管理者
add	コネクターの追加	許可	管理者
modify	コネクターの更新	許可	管理者
remove	コネクターの削除	許可	管理者

D.7.6. certServer.tps.groups

管理者がグループ操作を実行できるようにします。

```
allow (execute) group="Administrators"
```

表D.79 certServer.tps.groups ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
execute	グループ操作の実行	許可	管理者

D.7.7. certServer.tps.users

管理者がユーザー操作を実行できるようにします。

```
allow (execute) group="Administrators"
```

表D.80 certServer.tps.users ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
execute	ユーザー操作の実行	許可	管理者

D.7.8. certServer.tps.profiles

管理者と TPS エージェントがプロファイルのステータスを読み取って変更できるようにします。ただし、プロファイルを追加、変更、および削除できるのは管理者だけです。

```
allow (read,change-status) group="Administrators" || group="TPS Agents" ; allow
(add,modify,remove) group="Administrators"
```

表D.81 certServer.tps.profiles ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	プロフィールを読む	許可	管理者、TPS エージェント
change-status	プロファイルのステータスの変更	許可	管理者、TPS エージェント
add	プロファイルの追加	許可	管理者
modify	プロファイルの更新	許可	管理者
remove	プロファイルの削除	許可	管理者

D.7.9. certServer.tps.profile-mappings

管理者のみがプロファイルマッピングにアクセスできることを制御します。

```
allow (read,change-status,add,modify,remove) group="Administrators"
```

表D.82 certServer.tps.users ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	プロファイルマッピングの読み取り	許可	管理者
change-status	プロファイルマッピングのステータスの変更	許可	管理者
add	プロファイルマッピングの追加	許可	管理者
modify	プロファイル設定の更新	許可	管理者
remove	プロファイル設定の削除	許可	管理者

D.7.10. certServer.tps.selftests

管理者のみがセルフテストにアクセスできることを制御します。

```
allow (read,execute) group="Administrators"
```

表D.83 certServer.tps.selftests ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	セルフテストの読み取り	許可	管理者
execute	セルフテストの実行	許可	管理者

D.7.11. certServer.tps.tokens

管理者、エージェント、および Operator がトークンを読み取ることができるように制御します。ただし、管理者のみがトークンを追加および削除でき、エージェントのみがトークンを変更できます。

```
allow (read) group="Administrators" || group="TPS Agents" || group="TPS Operators"; allow (add,remove) group="Administrators" ; allow (modify) group="TPS Agents"
```

表D.84 certServer.tps.tokens ACL Summary

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	トークンの読み取り	許可	管理者、TPS エージェント、TPS Operator
add	トークンの追加	許可	管理者
remove	トークンの削除	許可	管理者
modify	トークンの更新	許可	TPS エージェント

付録E 監査イベント

この付録には 2 つの部分が含まれています。最初の部分、「必要な監査イベントと例」 には、CA プロテクションプロファイル V2.1 の要件 ID でグループ化された必要な監査イベントのリストが含まれています。各監査イベントには 1 つ以上の例が付随しています。次の部分、「監査イベントの説明」は、個別の監査イベントとそのパラメーターの説明および形式を提供します。ログ内のすべての監査イベントは、以下の情報を反映しています。

- スレッドの Java 識別子。以下に例を示します。

  > 0.localhost-startStop-1

- イベントが発生したタイムスタンプ。以下に例を示します。

  > [21/Jan/2019:17:53:00 IST]

- ログソース (14 は SIGNED_AUDIT)。

  > [14]

- 現在のログレベル (6 はセキュリティー関連のイベント)。『Red Hat Certificate System Planning, Installation, and Deployment Guide (Common Criteria Edition)』 の 『Log Levels (Message Categories)』 セクションを参照してください。以下に例を示します。

  > [6]

- ログイベントに関する情報 (ログイベント固有です。特定のログイベントの各フィールドに関する情報は 「監査イベントの説明」 を参照してください。以下に例を示します。

  > [AuditEvent=AUDIT_LOG_STARTUP][SubjectID=$System$][Outcome=Success] audit function startup

E.1. 必要な監査イベントと例

このセクションには、共通条件 CA 保護プロファイル v.2.1 ごとの必要なすべての監査イベントが含まれます。

監査イベントの説明は、「監査イベントの説明」 を参照してください。

FAU_GEN.1

- TSF 監査関数の起動
  - **AUDIT_LOG_STARTUP**

    > 0.localhost-startStop-1 - [21/Jan/2019:17:53:00 IST] [14] [6] [AuditEvent=AUDIT_LOG_STARTUP][SubjectID=$System$][Outcome=Success] audit function startup

- TFS インターフェイスを介して呼び出されるすべての管理アクション
  - **CONFIG_CERT_PROFILE**

> 0.http-bio-20443-exec-35 - [02/Jan/2019:05:05:09 EST] [14] [6] [AuditEvent=CONFIG_CERT_PROFILE][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;rules+Operation;;OP_ADD+Resource;;caAgentExample+class_id;;caEnrollImpl+name;;caAgentExample Enrollment Profile+description;;This certificate profile is for enrolling user certificates+visible;;true] certificate profile configuration parameter(s) change

- **CERT_PROFILE_APPROVAL**

> 0.http-bio-8443-exec-8 - [15/Nov/2018:15:37:19 PST] [14] [6] [AuditEvent=CERT_PROFILE_APPROVAL][SubjectID=cfuEC-0830-agent-2] [Outcome=Success][ProfileID=caTPSCert][Op=disapprove] certificate profile approval

- **CONFIG_OCSP_PROFILE**

> 0.http-bio-22443-exec-11 - [30/Jan/2019:06:18:02 EST] [14] [6] [AuditEvent=CONFIG_OCSP_PROFILE][SubjectID=ocspadmin][Outcome=Success] [ParamNameValPairs=Scope;;ocspStoresRules+Operation;;OP_MODIFY+Resource;;ldapStore+includeNextUpdate;;false+byName;;true+implName;;com.netscape.cms.ocsp.LDAPStore+numConns;;0+caCertAttr;;cACertificate;binary+notFoundAsGood;;true+crlAttr;;certificateRevocationList;binary] OCSP profile configuration parameter(s) change

- **CONFIG_CRL_PROFILE**

> 0.http-bio-20443-exec-48 - [29/Jan/2019:04:29:29 EST] [14] [6] [AuditEvent=CONFIG_CRL_PROFILE][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;crl+Operation;;OP_MODIFY+Resource;;MasterCRL+enableCRLUpdates;;true+updateSchema;;1+extendedNextUpdate;;true+alwaysUpdate;;false+enableDailyUpdates;;true+dailyUpdates;;4:30+enableUpdateInterval;;true+autoUpdateInterval;;240+nextUpdateGracePeriod;;0+nextAsThisUpdateExtension;;0] CRL profile configuration parameter(s) change

- **CONFIG_AUTH**

> 0.http-bio-20443-exec-11 - [15/Jan/2019:08:36:39 EST] [14] [6] [AuditEvent=CONFIG_AUTH][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;instance+Operation;;OP_ADD+Resource;;plug502+implName;;UidPwdDirAuth+ldap.ldapconn.host;;server.example.com+dnpattern;;uid=test,ou=people,o=topology-02-CA+ldapStringAttributes;;mail+ldap.ldapconn.version;;3+ldap.ldapconn.port;;3389+ldap.maxConns;;10+ldap.basedn;;dc=example,dc=com+ldap.minConns;;3+ldap.ldapconn.secureConn;;false+ldapByteAttributes;;uid+ldap.password;;(sensitive)+ldap.ldapauth.authtype;;BasicAuth+ldap.ldapauth.bindDN;;cn=direcory manager] authentication configuration parameter(s) change

> 0.http-bio-20080-exec-25 - [29/Jan/2019:04:54:14 EST] [14] [6] [AuditEvent=CONFIG_AUTH][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;instance+Operation;;OP_ADD+Resource;;plug7487+implName;;AgentCertAuth] authentication configuration parameter(s) change

- **CONFIG_ROLE(success)**

> 0.http-bio-20443-exec-50 - [18/Jan/2019:04:08:45 EST] [14] [6] [AuditEvent=CONFIG_ROLE][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;certs+Operation;;OP_ADD+Resource;;CA_AdminV+cert;;-----BEGIN CERTIFICATE-----MIIDYTCCAkmgAwIBAgIBfz...-----END CERTIFICATE-----] role configuration parameter(s) change

- **CONFIG_ROLE(Failure)**

> 0.http-bio-20443-exec-39 - [18/Jan/2019:04:08:57 EST] [14] [6] [AuditEvent=CONFIG_ROLE][SubjectID=caadmin][Outcome=Failure] [ParamNameValPairs=Scope;;users+Operation;;OP_ADD+Resource;;CA_AdminUnTrusted+password;;********+phone;;<null>+fullname;;CA_AdminUnTrusted+state;;<null>+userType;;<null>+email;;<null>] role configuration parameter(s) change

- **CONFIG_ACL**
  - CA

> CA = 0.http-bio-20443-exec-18 - [29/Jan/2019:05:15:16 EST] [14] [6] [AuditEvent=CONFIG_ACL][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Scope;;acls+Operation;;OP_MODIFY+Resource;;testACL+aci;;allow (read,allow) group="testGroup"+desc;;ALLOW READ to testGroup+rights;;read,allow] ACL configuration parameter(s) change

- **CONFIG_SIGNED_AUDIT**
  - CA

> 0.http-bio-20443-exec-20 - [29/Jan/2019:02:44:04 EST] [14] [6] [AuditEvent=CONFIG_SIGNED_AUDIT][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=Action;;disable] signed audit configuration parameter(s) change

  - KRA

> 0.http-bio-21443-exec-9 - [30/Jan/2019:08:15:11 EST] [14] [6] [AuditEvent=CONFIG_SIGNED_AUDIT][SubjectID=kraadmin][Outcome=Success] [ParamNameValPairs=Action;;enable] signed audit configuration parameter(s) change

  - OCSP

> 0.http-bio-22443-exec-17 - [30/Jan/2019:08:17:06 EST] [14] [6] [AuditEvent=CONFIG_SIGNED_AUDIT][SubjectID=ocspadmin][Outcome=Success] [ParamNameValPairs=Action;;enable] signed audit configuration parameter(s) change

  - TKS

> 0.http-bio-23443-exec-15 - [30/Jan/2019:08:18:52 EST] [14] [6] [AuditEvent=CONFIG_SIGNED_AUDIT][SubjectID=tksadmin][Outcome=Success] [ParamNameValPairs=Action;;enable] signed audit configuration parameter(s) change

- TPS

  > 0.http-bio-25443-exec-5 - [30/Jan/2019:08:20:03 EST] [14] [6]
  > [AuditEvent=CONFIG_SIGNED_AUDIT][SubjectID=tpsadmin][Outcome=Success]
  > [ParamNameValPairs=Action;;enable] signed audit configuration parameter(s)
  > change

- **CONFIG_TRUSTED_PUBLIC_KEY**

  - CA

    > 0.http-bio-20443-exec-9 - [29/Jan/2019:03:25:02 EST] [14] [6]
    > [AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY][SubjectID=caadmin]
    > [Outcome=Success]
    > [ParamNameValPairs=Scope;;installCert+Operation;;OP_MODIFY+Resource;;trusted
    > CACert+pkcs10;;-----BEGIN CERTIFICATE-----MIIEBDCCAuygAwI...-----END
    > CERTIFICATE-----+nickname;;<null>+pathname;;<null>+serverRoot;;
    > <null>+serverID;;instanceID] certificate database configuration

  - KRA

    > 0.http-bio-21443-exec-17 - [30/Jan/2019:08:29:07 EST] [14] [6]
    > [AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY][SubjectID=kraadmin]
    > [Outcome=Success]
    > [ParamNameValPairs=Scope;;installCert+Operation;;OP_MODIFY+Resource;;trusted
    > CACert+pkcs10;;-----BEGIN CERTIFICATE-----MIIEBDCCAuygAw...-----END
    > CERTIFICATE-----+nickname;;<null>+pathname;;<null>+serverRoot;;
    > <null>+serverID;;instanceID] certificate database configuration

  - OCSP

    > 0.http-bio-22443-exec-25 - [30/Jan/2019:08:41:08 EST] [14] [6]
    > [AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY][SubjectID=ocspadmin]
    > [Outcome=Success]
    > [ParamNameValPairs=Scope;;installCert+Operation;;OP_MODIFY+Resource;;trusted
    > CACert+pkcs10;;-----BEGIN CERTIFICATE-----MIIEBDCCAuygAwIB...-----END
    > CERTIFICATE-----+nickname;;<null>+pathname;;<null>+serverRoot;;
    > <null>+serverID;;instanceID] certificate database configuration

  - TKS

    > 0.http-bio-23443-exec-23 - [30/Jan/2019:08:45:40 EST] [14] [6]
    > [AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY][SubjectID=tksadmin]
    > [Outcome=Success]
    > [ParamNameValPairs=Scope;;installCert+Operation;;OP_MODIFY+Resource;;trusted
    > CACert+pkcs10;;-----BEGIN CERTIFICATE-----MIIEBDCCAuygAwIBA...-----END
    > CERTIFICATE-----+nickname;;<null>+pathname;;<null>+serverRoot;;
    > <null>+serverID;;instanceID] certificate database configuration

  - TPS

    > 0.http-bio-22443-exec-23 - [30/Jan/2019:08:46:13 EST] [14] [6]

> [AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY][SubjectID=tpsadmin] [Outcome=Success] [ParamNameValPairs=Scope;;installCert+Operation;;OP_MODIFY+Resource;;trusted CACert+pkcs10;;-----BEGIN CERTIFICATE-----MIIEBDCCAuygAwIBA...-----END CERTIFICATE-----+nickname;;<null>+pathname;;<null>+serverRoot;; <null>+serverID;;instanceID] certificate database configuration

- **CONFIG_DRM**

> 0.http-bio-21443-exec-1 - [24/Jan/2019:09:36:52 EST] [14] [6] [AuditEvent=CONFIG_DRM][SubjectID=kraadmin][Outcome=Success] [ParamNameValPairs=Scope;;general+Operation;;OP_MODIFY+Resource;;RS_ID_CON FIG+noOfRequiredRecoveryAgents;;2] DRM configuration parameter(s) change

- **OCSP_ADD_CA_REQUEST_PROCESSED**
  - 正常に接続できる場合

> 0.http-bio-22443-exec-24 - [29/Jan/2019:03:15:59 EST] [14] [6] [AuditEvent=OCSP_ADD_CA_REQUEST_PROCESSED][SubjectID=ocspadmin] [Outcome=Success][CASubjectDN=CN=CA Signing Certificate,OU=topology-02-CA,O=topology-02_example.com] Add CA for OCSP Responde

  - 失敗

> 0.http-bio-22443-exec-12 - [30/Jan/2019:06:44:32 EST] [14] [6] [AuditEvent=OCSP_ADD_CA_REQUEST_PROCESSED][SubjectID=ocspadmin] [Outcome=Failure][CASubjectDN=<null>] Add CA for OCSP Responder

- **OCSP_REMOVE_CA_REQUEST_PROCESSED**

> 0.http-bio-22443-exec-24 - [29/Jan/2019:03:13:43 EST] [14] [6] [AuditEvent=OCSP_REMOVE_CA_REQUEST_PROCESSED][SubjectID=ocspadmin] [Outcome=Success][CASubjectDN=CN=CA Signing Certificate,OU=topology-02-CA,O=topology-02_example.com] Remove CA for OCSP Responder is successful

- **SECURITY_DOMAIN_UPDATE**
  - Operation: Issue_token

> 0.http-bio-20443-exec-10 - [16/Jan/2019:03:19:57 EST] [14] [6] [AuditEvent=SECURITY_DOMAIN_UPDATE][SubjectID=caadmin] [Outcome=Success] [ParamNameValPairs=operation;;issue_token+token;;2433856184928074456+ip;;192. 0.2.1+uid;;caadmin+groupname;;Enterprise TKS Administrators] security domain update

  - Operation: Add

> 0.http-bio-20443-exec-18 - [02/Jan/2019:04:39:21 EST] [14] [6] [AuditEvent=SECURITY_DOMAIN_UPDATE][SubjectID=caadmin] [Outcome=Success][ParamNameValPairs=host;;server.example.com+name;;OCSP

> server.example.com 22443+sport;;22443+clone;;false+type;;OCSP+operation;;add] security domain update

- **CONFIG_SERIAL_NUMBER**
  - CA

> 0.http-bio-20443-exec-2 - [29/Jan/2019:07:53:21 EST] [14] [6] [AuditEvent=CONFIG_SERIAL_NUMBER][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=source;;updateNumberRange+type;;request+beginNumber;;9990001+endNumber;;10000000] serial number range update

  - KRA

> 0.http-bio-21443-exec-7 - [18/Jan/2019:19:11:47 EST] [14] [6] [AuditEvent=CONFIG_SERIAL_NUMBER][SubjectID=caadmin][Outcome=Success] [ParamNameValPairs=source;;updateNumberRange+type;;serialNo+beginNumber;;fff0001+endNumber;;10000000] serial number range update

FDP_CER_EXT.1 (extended)

- 証明書の生成
  - **CERT_REQUEST_PROCESSED (SUCCESS)**

> 0.http-bio-8443-exec-24 - [07/Sep/2018:10:21:57 PDT] [14] [6] [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=caadmin][Outcome=Success] [ReqID=7][CertSerialNum=7] certificate request processed

FDP_CER_EXT.2 (拡張)

- 証明書要求への証明書のリンク
  - **PROFILE_CERT_REQUEST**

> 0.http-bio-8443-exec-24 - [07/Sep/2018:10:21:57 PDT] [14] [6] [AuditEvent=PROFILE_CERT_REQUEST][SubjectID=caadmin][Outcome=Success] [ReqID=7][ProfileID=caECFullCMCUserCert][CertSubject=CN=cfuEC-0830] certificate request made with certificate profiles

注記

ReqID フィールドは、成功した **CERT_REQUEST_PROCESSED** イベントの ***ReqID*** フィールドに効果的にリンクします。

FDP_CER_EXT.3

- 失敗した証明書の承認
  - **CERT_REQUEST_PROCESSED (FAILURE)**

> 0.http-bio-20443-exec-4 - [21/Jan/2019:00:24:16 EST] [14] [6] [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=$NonRoleUser$]

> [Outcome=Failure][ReqID=1483][InfoName=rejectReason][InfoValue=Request 1483 Rejected - Subject Name Not Matched UID=testuser00,E=example@example.com,CN=MyTestUser] certificate request processed

FIA_X509_EXT.1, FIA_X509_EXT.2

- 証明書の検証の失敗、認証の失敗
  - **ACCESS_SESSION_ESTABLISH (FAILURE)**
    - 証明書が取り消されたユーザーが操作を実行しようとしています。

      > 0.http-bio-21443-exec-9 - [12/Feb/2019:14:52:26 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2] [SubjectID=UID=KRA_AgentR,E=KRA_AgentR@example.org,CN=KRA_AgentR,OU=IDMQE,C=US][Outcome=Failure][Info=CERTIFICATE_REVOKED] access session establish failure

    - 期限切れの証明書を持つユーザーが操作を実行しようとしています。

      > 0.http-bio-21443-exec-9 - [12/Feb/2019:14:52:26 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2] [SubjectID=UID=KRA_AgentR,E=KRA_AgentR@example.org,CN=KRA_AgentR,OU=IDMQE,C=US][Outcome=Failure][Info=CERTIFICATE_EXPIRED] access session establish failure

    - 不明な CA によって発行された TLS クライアント証明書を使用して送信された CMC 登録要求。

      > 0.http-bio-20443-exec-28 - [12/Feb/2019:16:31:08 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2][SubjectID=CN=CA Signing Certificate,OU=pki-tomcat,O=EXAMPLE][Outcome=Failure][Info=UNKNOWN_CA] access session establish failure

    - クライアントのプロトコルが一致しない場合。たとえば、クライアントは **ssl3** を使用しますが、サーバーはサポートしていません。

      > 0.http-bio-20443-exec-11 - [12/Feb/2019:16:35:26 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2][SubjectID=][Outcome=Failure][Info=HANDSHAKE_FAILURE] access session establish failure

    - プロトコルバージョンが正しくありません。サンプルサーバーは **tls1.1 および tls1.** 2 をサポートしますが、クライアントは **tls1** を送信します。

      > 0.http-bio-20443-exec-46 - [12/Feb/2019:16:39:10 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2][SubjectID=][Outcome=Failure][Info=PROTOCOL_VERSION] access session establish failure

- クライアントが暗号の一覧を送信しますが、サーバーには暗号の一覧がありません。

  サーバー

  > 0.http-bio-21443-exec-3 - [13/Feb/2019:07:40:44 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2][SubjectID=][Outcome=Failure][Info=INTERNAL_ERROR] access session establish failure

FIA_UIA_EXT.1

- 特権ユーザーの識別と認証
  - **ACCESS_SESSION_ESTABLISH**
    - CA の例

      > 0.http-bio-8443-exec-1 - [10/Oct/2018:15:42:13 PDT] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=][Outcome=Success] access session establish success

    - TPS の例

      > 0.http-bio-25443-exec-1 - [02/Jan/2019:04:44:12 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=][Outcome=Success] access session establish success

  - **AUTH**
    - CA の例

      > 0.http-bio-8443-exec-1 - [28/Nov/2018:16:23:15 PST] [14] [6] [AuditEvent=AUTH] [SubjectID=caagentJoe][Outcome=Success][AuthMgr=CMCAuth] authentication success

    - TPS の例

      > 0.http-bio-25443-exec-1 - [25/Jan/2019:13:00:59 IST] [14] [6] [AuditEvent=AUTH] [SubjectID=tpsadmin][Outcome=Success][AuthMgr=passwdUserDBAuthMgr] authentication success

  - **AUTHZ**
    - CA の例

      > 0.http-bio-8443-exec-1 - [28/Nov/2018:16:23:15 PST] [14] [6] [AuditEvent=AUTHZ] [SubjectID=caagentJoe][Outcome=Success][aclResource=certServer.ee.profile] [Op=submit] authorization success

    - TPS の例

> 0.http-bio-25443-exec-1 - [25/Jan/2019:13:00:59 IST] [14] [6] [AuditEvent=AUTHZ] [SubjectID=tpsadmin][Outcome=Success][aclResource=certServer.tps.account] [Op=login][Info=AccountResource.login] authorization success

- **ROLE_ASSUME**
  - CA の例

    > 0.http-bio-8443-exec-1 - [28/Nov/2018:16:23:15 PST] [14] [6] [AuditEvent=ROLE_ASSUME][SubjectID=caagentJoe][Outcome=Success] [Role=Certificate Manager Agents] assume privileged role

  - TPS の例

    > 0.http-bio-25443-exec-9 - [25/Jan/2019:13:00:07 IST] [14] [6] [AuditEvent=ROLE_ASSUME][SubjectID=cfu][Outcome=Success][Role=Certificate Manager Agents] assume privileged role

FMT_SMR.2

- ロールの一部であるユーザーのグループへの変更
  - **CONFIG_ROLE**

    上記の **CONFIG_ROLE** イベントを参照してください。

FPT_FLS.1

- セキュア状態の保持による失敗
  - **SELFTESTS_EXECUTION**
    - CA の例

      > 0.localhost-startStop-1 - [10/Jan/2019:00:47:57 EST] [14] [6] [AuditEvent=SELFTESTS_EXECUTION][SubjectID=$System$][Outcome=Failure] self tests execution (see selftests.log for details)

    - TPS の例

      > 0.localhost-startStop-1 - [22/Jan/2019:11:55:32 IST] [14] [6] [AuditEvent=SELFTESTS_EXECUTION][SubjectID=$System$][Outcome=Failure] self tests execution (see selftests.log for details)

FPT_KST_EXT.2

- 秘密鍵/秘密鍵は HSM によって格納され、これらの鍵にアクセスする唯一の操作は、署名操作として TSF を使用することです。

  > CERT_REQUEST_PROCESSED (failure)
  > 0.http-bio-20443-exec-8 - [28/Jan/2019:13:48:14 EST] [14] [6] [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=$Unidentified$][Outcome=Failure] [ReqID=28][InfoName=rejectReason][InfoValue=Request Key Type RSA Not Matched Rejected - {1}] certificate request processed

FPT_RCV.1

- 障害またはサービスの中断が発生したという事実。通常の操作の再開
  - 失敗： **SELFTESTS_EXECUTION （失敗）**
    - CA の例

      0.localhost-startStop-1 - [29/Jan/2019:13:29:03 UTC] [14] [6] [AuditEvent=SELFTESTS_EXECUTION][SubjectID=$System$][Outcome=Failure] self tests execution (see selftests.log for details)

    - TPS の例

      0.localhost-startStop-1 - [22/Jan/2019:11:55:32 IST] [14] [6] [AuditEvent=SELFTESTS_EXECUTION][SubjectID=$System$][Outcome=Failure] self tests execution (see selftests.log for details)

  - セルフテストのログは 13.3.2 Red Hat Certificat Systemitem の計画、インストール、およびデプロイメントのガイドのセルフテストの設定を参照してください。
  - 再開： **AUDIT_LOG_STARTUP、SELFTESTS_EXECUTION （成功）**
    - TPS の例

      0.localhost-startStop-1 - [21/Jan/2019:16:47:44 IST] [14] [6] [AuditEvent=AUDIT_LOG_STARTUP][SubjectID=$System$][Outcome=Success] audit function startup

    - CA の例

      0.localhost-startStop-1 - [04/Feb/2019:18:29:38 EST] [14] [6] [AuditEvent=SELFTESTS_EXECUTION][SubjectID=$System$][Outcome=Success] self tests execution (see selftests.log for details)

FPT_STM.1

- 時間の変更。

  「時間変更イベントの表示」 を参照してください

FPT_TUD_EXT.1

- 更新の開始。

  「パッケージ更新イベントの表示」を参照してください。

FTA_SSL.4

- 対話セッションの終了。
  - **ACCESS_SESSION_TERMINATED**
    - 0.http-bio-20443-exec-7 - [21/Jan/2019:03:42:17 EST] [14] [6] [AuditEvent=ACCESS_SESSION_TERMINATED][ClientIP=192.0.2.1]

> [ServerIP=192.0.2.1][SubjectID=CN=PKI Administrator,E=caadmin@example.com,OU=topology-02-CA,O=topology-02_example.com][Outcome=Success][Info=CLOSE_NOTIFY] access session terminated

- TPS

> 0.http-bio-25443-exec-1 - [02/Jan/2019:04:44:12 EST] [14] [6] [AuditEvent=ACCESS_SESSION_TERMINATED][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=][Outcome=Success][Info=CLOSE_NOTIFY] access session

FTP_TRP.1

- 信頼できるチャンネルの開始信頼できるチャンネルの終了信頼できるパス関数の失敗
  - **ACCESS_SESSION_ESTABLISH**
    - > 2529:0.http-bio-20443-exec-8 - [29/Jan/2019:02:41:10 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=CN=PKI Administrator,E=tpsadmin@server.example.com,OU=topology-02-TPS,O=topology-02_example.com][Outcome=Failure][Info=UNKNOWN_CA] access session establish failure
    - TPS
    - > 0.http-bio-25443-exec-4 - [25/Jan/2019:12:58:31 IST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=0:0:0:0:0:0:0:1] [ServerIP=0:0:0:0:0:0:0:1][SubjectID=][Outcome=Failure] [Info=RECORD_OVERFLOW] access session establish failure
  - **ACCESS_SESSION_TERMINATED**
    - > 0.http-bio-20443-exec-48 - [29/Jan/2019:04:30:49 EST] [14] [6] [AuditEvent=ACCESS_SESSION_TERMINATED][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=][Outcome=Success][Info=CLOSE_NOTIFY] access session terminated
    - TPS
    - > TPS=0.http-bio-25443-exec-19 - [25/Jan/2019:12:47:07 IST] [14] [6] [AuditEvent=ACCESS_SESSION_TERMINATED][ClientIP=192.0.2.1] [ServerIP=192.0.2.1][SubjectID=][Outcome=Success][Info=CLOSE_NOTIFY] access session terminated

FCS_CKM.1 および FCS_CKM.2

- 利用できません。TOE サブシステムが非一時キーを生成する (または OE に生成を要求する) TOE 関連の機能はありません。すべてのシステム証明書は、インストール中、つまり TOE が実行される前、つまり TOE が監査できるようになる前に、ユーザーキーと同じ方法で生成されます。

FCS_CKM_EXT.4

- 利用不可

FCS_COP.1(2)

- CA 署名キーを使用した署名生成のすべての発生。
  - **CERT_SIGNING_INFO** は、システムの起動時に CA 署名証明書のキー情報を記録します。

    > 0.authorityMonitor - [03/Jan/2019:02:33:35 EST] [14] [6]
    > [AuditEvent=CERT_SIGNING_INFO][SubjectID=$System$][Outcome=Success]
    > [SKI=E3:D2:5B:2A:F5:76:FF:7B:48:CA:94:18:5F:7B:BD:6B:95:FB:8F:30]
    > [AuthorityID=dbec10a4-1264-4759-96d5-6d2aadbf9d34] certificate signing info

  - **CERT_REQUEST_PROCESSED (成功)**

    > 0.http-bio-20443-exec-378 - [19/Jan/2019:05:57:39 EST] [14] [6]
    > [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=caadmin][Outcome=Success]
    > [ReqID=1352][CertSerialNum=984] certificate request processed

  - **OCSP_SIGNING_INFO** は、システムの起動時に OCSP 署名証明書キー情報を記録します

    > 0.http-bio-29443-exec-3 - [10/Oct/2018:14:15:24 PDT] [14] [6]
    > [AuditEvent=OCSP_SIGNING_INFO][SubjectID=$System$][Outcome=Success]
    > [SKI=71:B1:D0:AE:44:DF:ED:D0:20:15:2B:E3:37:E8:EE:04:EB:D6:F1:44] OCSP signing
    > info

  - **OCSP_GENERATION (成功)**

    > 0.http-nio-22080-exec-3 - [31/Jan/2019:15:34:47 EST] [14] [6]
    > [AuditEvent=OCSP_GENERATION][SubjectID=$NonRoleUser$][Outcome=Success]
    > OCSP response generation

  - **CRL_SIGNING_INFO** は、システムの起動時に CRL 署名証明書キー情報を記録します

    > 0.localhost-startStop-1 - [10/Jan/2019:09:10:27 EST] [14] [6]
    > [AuditEvent=CRL_SIGNING_INFO][SubjectID=$System$][Outcome=Success]
    > [SKI=23:98:ED:52:5B:2C:27:C6:FF:7C:34:D1:D5:48:57:E9:B8:D1:4E:95] CRL signing
    > info

  - **FULL_CRL_GENERATION (成功)**

    > 0.CRLIssuingPoint-testing123 - [30/Jan/2019:08:35:02 EST] [14] [6]
    > [AuditEvent=FULL_CRL_GENERATION][SubjectID=$System$][Outcome=Success]
    > [CRLnum=6] Full CRL generation

  - **DELTA_CRL_GENERATION (成功)**

    > 0.CRLIssuingPoint-testing123 - [30/Jan/2019:08:35:01 EST] [14] [6]
    > [AuditEvent=DELTA_CRL_GENERATION][SubjectID=$Unidentified$]
    > [Outcome=Success][CRLnum=5] Delta CRL generation

- 署名生成の失敗。
  - **CERT_REQUEST_PROCESSED (失敗)**

    > 0.http-bio-20443-exec-8 - [28/Jan/2019:13:48:14 EST] [14] [6] [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=$Unidentified$] [Outcome=Failure][ReqID=28][InfoName=rejectReason][InfoValue=Request Key Type RSA Not Matched Rejected - {1}] certificate request processed

  - **OCSP_GENERATION (失敗)**

    > 0.http-nio-22080-exec-6 - [31/Jan/2019:15:35:38 EST] [14] [6] [AuditEvent=OCSP_GENERATION][SubjectID=$NonRoleUser$][Outcome=Failure] [FailureReason=Missing issuer certificate] OCSP response generation

  - **FULL_CRL_GENERATION (失敗)**

FCS_HTTPS_EXT.1 および FCS_TLSS_EXT.2

- HTTPS/TLS セッションを確立できません。
  - **ACCESS_SESSION_ESTABLISH (失敗)**

    > See FTP_TRP.1

- HTTPS/TLS セッションの確立/判断
  - **ACCESS_SESSION_TERMINATED**

    > See FIA_UIA_EXT.1

FCS_TLSC_EXT.2

- TLS セッションを確立できません。
  - **CLIENT_ACCESS_SESSION_ESTABLISH (失敗)**

    > 0.http-bio-20443-exec-21 - [13/Feb/2019:07:48:08 EST] [14] [6] [AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH][ClientHost=192.0.2.1] [ServerHost=pki1.example.com][ServerPort=21443][SubjectID=SYSTEM] [Outcome=Failure][Info=send:java.io.IOException: SocketException cannot write on socket] access session failed to establish when Certificate System acts as client

    クライアントがサーバーに到達できず、セッションが失敗した場合。このシナリオでは、CA はキーアーカイブ中に KRA のクライアントとして機能し、KRA は CA から到達できません。

    > 0.http-bio-20443-exec-11 - [12/Feb/2019:18:20:03 EST] [14] [6] [AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH][ClientHost=192.0.2.1] [ServerHost=pki1.example.com][ServerPort=21443][SubjectID=SYSTEM] [Outcome=Failure][Info=send:java.io.IOException: Socket has been closed, and cannot be reused.] access session failed to establish when Certificate System acts as client

    CA のサブシステム証明書が取り消され、KRA にアクセスしようとした場合。

- KRA

  > 0.http-bio-21443-exec-3 - [13/Feb/2019:08:15:53 EST] [14] [6] [AuditEvent=ACCESS_SESSION_ESTABLISH][ClientIP=192.0.2.1] [ServerIP=192.0.2.2][SubjectID=CN=Subsystem Certificate,OU=topology-02-CA,O=topology-02_Foobarmaster.org][Outcome=Failure] [Info=CERTIFICATE_REVOKED] access session establish failure

- CA

  > 0.http-bio-20443-exec-10 - [13/Feb/2019:08:16:08 EST] [14] [6] [AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH][ClientHost=192.0.2.1] [ServerHost=pki1.example.com][ServerPort=21443][SubjectID=SYSTEM] [Outcome=Failure][Info=send:java.io.IOException: SocketException cannot write on socket] access session failed to establish when Certificate System acts as client

- TLS セッションの確立/終了。
  - **CLIENT_ACCESS_SESSION_TERMINATED**

    > 0.http-bio-8443-exec-6 - [10/Oct/2018:15:10:54 PDT] [14] [6] [AuditEvent=CLIENT_ACCESS_SESSION_TERMINATED][ClientHost=192.0.2.1] [ServerHost=192.0.2.1][ServerPort=29443][SubjectID=SYSTEM][Outcome=Success] [Info=CLOSE_NOTIFY] access session terminated when Certificate System acts as client

FDP_CRL_EXT.1

- CRL の生成に失敗
  - **FULL_CRL_GENERATION (失敗)**

    > 0.http-bio-20444-exec-9 - [01/Feb/2019:15:40:38 EST] [14] [6] [AuditEvent=FULL_CRL_GENERATION][SubjectID=caadmin][Outcome=Failure] [FailureReason=Record not found] Full CRL generation

FDP_OCSPG_EXT.1

- 証明書のステータス情報の生成に失敗
  - **OCSP_GENERATION (失敗)**

FIA_AFL.1

- Unsuccessful Authentication Attempts のしきい値に達しています。実行されたアクション。無効化された非管理アカウントの再有効化。

  利用できません。パスワード認証のみ。Certificate System は、証明書ベースの認証のみを提供します。

FIA_CMCS_EXT.1

- 証明書要求または失効要求を含む CMC 要求 (生成または受信)。CMC の応答が発行されます。
  - **CMC_SIGNED_REQUEST_SIG_VERIFY**

> 0.http-bio-20080-exec-22 - [24/Jan/2019:08:44:51 EST] [14] [6]
> [AuditEvent=CMC_SIGNED_REQUEST_SIG_VERIFY][SubjectID=$NonRoleUser$]
> [Outcome=Failure][ReqType=$Unidentified$][CertSubject=$Unidentified$]
> [SignerInfo=$Unidentified$] agent signed CMC request signature verification

- **CMC_USER_SIGNED_REQUEST_SIG_VERIFY**
  - 成功したリクエスト:

> 0.http-bio-20443-exec-1 - [18/Feb/2019:12:07:20 EST] [14] [6]
> [AuditEvent=CMC_USER_SIGNED_REQUEST_SIG_VERIFY]
> [SubjectID=UID=test10,CN=test10,O=example.org][Outcome=Success]
> [ReqType=enrollment][CertSubject=<null>]
> [SignerInfo=UID=test10,CN=test10,O=example.org] User signed CMC request
> signature verification success

- **CMC_REQUEST_RECEIVED**
  - 成功したリクエスト:

> 0.http-bio-20443-exec-13 - [29/Jan/2019:04:26:49 EST] [14] [6]
> [AuditEvent=CMC_REQUEST_RECEIVED][SubjectID=$Unidentified$]
> [Outcome=Success][CMCRequest=MIICoAYJKoZIhv...] CMC request received

  - 失敗したリクエスト:

> 0.http-bio-20443-exec-14 - [29/Jan/2019:07:15:27 EST] [14] [6]
> [AuditEvent=CMC_REQUEST_RECEIVED][SubjectID=$Unidentified$]
> [Outcome=Success][CMCRequest=MIGOBgkqhkiG9w...] CMC request received

- **PROOF_OF_POSSESSION** (Enrollment Event)

> 0.http-bio-20443-exec-13 - [29/Jan/2019:04:26:49 EST] [14] [6]
> [AuditEvent=PROOF_OF_POSSESSION][SubjectID=user1a][Outcome=Success]
> [Info=method=EnrollProfile: verifyPOP:] proof of possession

- **PROFILE_CERT_REQUEST** (Enrollment Event)

> 0.http-bio-20443-exec-13 - [29/Jan/2019:04:26:49 EST] [14] [6]
> [AuditEvent=PROFILE_CERT_REQUEST][SubjectID=user1a][Outcome=Success]
> [ReqID=31][ProfileID=caECFullCMCSharedTokenCert]
> [CertSubject=UID=user1a,OU=People,DC=rhel76,DC=test] certificate request made with
> certificate profiles

- **CERT_STATUS_CHANGE_REQUEST**
  - 成功:

> 0.http-bio-20443-exec-5 - [05/Feb/2019:05:57:12 EST] [14] [6]
> [AuditEvent=CERT_STATUS_CHANGE_REQUEST][SubjectID=caadmin]
> [Outcome=Success][ReqID=121][CertSerialNum=0x67][RequestType=on-hold]
> certificate revocation/unrevocation request made

- 失敗:

> 0.http-bio-20443-exec-13 - [05/Feb/2019:05:58:55 EST] [14] [6] [AuditEvent=CERT_STATUS_CHANGE_REQUEST][SubjectID=caadmin] [Outcome=Failure][ReqID=<null>][CertSerialNum=0x67][RequestType=on-hold] certificate revocation/unrevocation request made

- **CERT_REQUEST_PROCESSED**
  - 成功したリクエスト:

> 0.http-bio-20443-exec-13 - [29/Jan/2019:04:26:49 EST] [14] [6] [AuditEvent=CERT_REQUEST_PROCESSED][SubjectID=$Unidentified$] [Outcome=Success][ReqID=31][CertSerialNum=20] certificate request processed

- **CERT_STATUS_CHANGE_REQUEST_PROCESSED**
  - 成功したリクエスト:

> 0.http-bio-20443-exec-9 - [29/Jan/2019:07:43:36 EST] [14] [6] [AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED] [SubjectID=UID=user1a,OU=People,DC=rhel76,DC=test][Outcome=Success] [ReqID=32][CertSerialNum=20][RequestType=revoke] [RevokeReasonNum=Certificate_Hold][Approval=complete] certificate status change request processed

  - 失敗したリクエスト:

> 0.http-bio-20443-exec-14 - [29/Jan/2019:07:15:27 EST] [14] [6] [AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED][SubjectID=<null>][Outcome=Failure][ReqID=<null>][CertSerialNum=20] [RequestType=revoke][RevokeReasonNum=Certificate_Hold] [Approval=rejected][Info=CMCOutputTemplate: SharedSecret.getSharedToken(BigInteger serial): shrTok not found in metaInfo] certificate status change request processed

> 0.http-bio-20443-exec-20 - [29/Jan/2019:07:30:41 EST] [14] [6] [AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED] [SubjectID=UID=user1a,OU=People,DC=rhel76,DC=test][Outcome=Failure] [ReqID=<null>][CertSerialNum=20][RequestType=revoke] [RevokeReasonNum=Certificate_Hold][Approval=rejected][Info= certificate issuer DN and revocation request issuer DN do not match] certificate status change request processed

> 0.http-bio-20443-exec-16 - [29/Jan/2019:07:55:27 EST] [14] [6] [AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED][SubjectID=<null>][Outcome=Failure][ReqID=<null>][CertSerialNum=20] [RequestType=revoke][RevokeReasonNum=Certificate_Hold] [Approval=rejected][Info= shared secret not found] certificate status change request processed

- **CMC_RESPONSE_SENT**

- 登録
  - 正常な応答

    0.http-bio-20443-exec-13 - [29/Jan/2019:04:26:49 EST] [14] [6] [AuditEvent=CMC_RESPONSE_SENT][SubjectID=user1a][Outcome=Success] [CMCResponse=MIIHTAYJKoZI...] CMC response sent

- 取り消し
  - 失効の成功

    0.http-bio-20443-exec-9 - [29/Jan/2019:07:43:36 EST] [14] [6] [AuditEvent=CMC_RESPONSE_SENT][SubjectID=$Unidentified$] [Outcome=Success][CMCResponse=MIIExgYJKoZ...] CMC response sent

  - 失効に失敗
    - 失効が行われない

      0.http-bio-20443-exec-20 - [29/Jan/2019:07:30:41 EST] [14] [6] [AuditEvent=CMC_RESPONSE_SENT][SubjectID=$Unidentified$] [Outcome=Success][CMCResponse=MIIFDgYJKoZIh...] CMC response sent

FPT_SKY_EXT.1(2)/OTH

- **AUTHZ**
  - 失敗: エージェントユーザーが監査ログの取得を試行

    0.http-bio-8443-exec-2 - [22/Feb/2019:15:03:38 PST] [14] [6] [AuditEvent=AUTHZ] [SubjectID=EC-CA-agent-2][Outcome=Failure] [aclResource=certServer.log.content.signedAudit][Op=read][Info=Authorization Error] authorization failure

  - 成功: 監査ユーザーが監査ログを取得:

    0.http-bio-8443-exec-13 - [22/Feb/2019:15:25:34 PST] [14] [6] [AuditEvent=AUTHZ] [SubjectID=EC-CA-auditor][Outcome=Success] [aclResource=certServer.log.content.signedAudit][Op=read] [Info=AuditResource.getAuditFile] authorization success

FTP_ITC.1

- 信頼できるチャンネルの開始信頼できるチャンネルの終了信頼できるチャンネル関数に失敗。
  - FCS_HTTPS_EXT.1 を参照
  - FCS_TLSC_EXT.2 を参照

E.2. 監査イベントの説明

必要な監査イベントとその例については、「必要な監査イベントと例」 を参照してください。

```
######################## 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)
```

用語集

A

administrator

1 つ以上の Certificate System マネージャーをインストールおよび設定し、それらの特権ユーザーまたはエージェントをセットアップする人。agent も併せて参照してください。

agent

Certificate System マネージャーで agent services の管理を許可されたグループに属するユーザー。Certificate Manager エージェント、キーリカバリー認証局エージェントも参照してください。

agent services

1.エージェントに必要な特権が割り当てられた Certificate System サブシステムが提供する HTML ページを通じて、Certificate System agent が管理できるサービス。

2.このようなサービスを管理するための HTML ページ。

agent-approved enrollment

証明書の発行前に、エージェントによる要求を承認するのに必要な登録。

APDU

アプリケーションプロトコルデータユニット。スマートカードとスマートカードリーダーとの間の通信に使用される通信ユニット (バイトに類似)。

auditor

署名付き監査ログを表示できる特権ユーザー。

アクセス制御

特定ユーザーが実行できるものを制御するプロセス。たとえば、サーバーへのアクセス制御は通常、パスワードまたは証明書によって確立された ID と、そのエンティティーが実行できることに関するルールに基づいています。アクセス制御リスト (ACL) も参照してください。

アクセス制御リスト (ACL)

サーバーが特定のリソースへのアクセス要求を受け取ったときに評価されるアクセスルールの階層を定義するアクセス制御エントリーのコレクション。アクセス制御手順 (ACI) を参照してください。

アクセス制御手順 (ACI)

アクセスを要求するサブジェクトを識別する方法、または特定のサブジェクトに対して許可または拒否される権限を指定するアクセスルール。アクセス制御リスト (ACL) を参照してください。

属性値アサーション (AVA)

attribute = value の形式のアサーションで、**attribute** は **o** (組織) または **uid** (ユーザー ID) などのタグ、**value** は Red Hat, Inc.やログイン名などの値です。AVA は、証明書の subject name と呼ばれる、証明書の賢明を識別する 識別名 (DN) を形成するために使用されます。

監査ログ

さまざまなシステムイベントを記録するログこのログは署名して、改ざんされなかった証明を提供でき、auditor ユーザーのみが読み取りできます。

自動登録

人の介入なしに、エンドエンティティー登録の自動認証を可能にする Certificate System サブシステムを設定する方法。この形式の認証では、認証モジュールの処理を正常に完了した証明書要求が、プロファイル処理と証明書の発行に対して自動的に承認されます。

認可

サーバーが制御するリソースにアクセスするパーミッション。承認は通常、リソースに関連付けられた ACL がサーバーによって評価された後に行われます。アクセス制御リスト (ACL) を参照してください。

認証

自信を持って識別すること。コンピューター化された送信の当事者が偽者ではないことを保証すること。認証には通常、パスワード、証明書、PIN、またはその他の情報を使用して、コンピューターネットワーク上で ID を検証することが含まれます。パスワードベースの認証、証明書ベースの認証、クライアント認証、サーバー認証 も参照してください。

認証モジュール

ルールのセット (として実装されますJava™クラス) エンドエンティティー、エージェント、管理者、または Certificate System サブシステムと対話する必要があるその他のエンティティーを認証するため。通常のエンドユーザー登録の場合は、ユーザーが登録フォームで要求された情報を入力した後、登録サーブレットはそのフォームに関連付けられた認証モジュールを使用して情報を検証し、ユーザーの ID を認証します。servlet を参照してください。

高度暗号化標準 (AES)

Advanced Encryption Standard (AES) は、その前身の Data Encryption Standard (DES) と同様に、FIPS 承認の対称鍵暗号化標準です。AES は 2002 年に米国政府によって採用されました。AES-128、AES-192、および AES-256 の 3 つのブロック暗号を定義します。NIST (National Institute of Standards and Technology) は、米国で AES 標準を定義しています。FIPS PUB 197。詳細は、http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf を参照してください。

B

bind DN

Red Hat Directory Server への認証にパスワードとともに使用される識別名 (DN) の形式のユーザー ID。

C

CA サーバーキー

CA サービスを提供するサーバーの TLS サーバーキー。

CA 署名鍵

CA 証明書の公開鍵に対応する秘密鍵。CA は、その署名キーを使用して証明書および CRL に署名します。

CA 証明書

認証局を識別する証明書。認証局 (CA)、subordinate CA、root CA も参照してください。

CA 階層

ルート CA が下位 CA に証明書を発行する権限を委任する CA の階層。下位 CA は、発行ステータスを他の CA に委譲して階層を拡張することもできます。認証局 (CA)、subordinate CA、root CA も参照してください。

certificate

X.509 標準に準拠してフォーマットされたデジタルデータで、個人、会社などのエンティティー名 (証明書の subject name) を指定し、証明書に含まれる 公開鍵 もそのエンティティーに属することを証明するもの。証明書は発行され、認証局 (CA) により署名されます。証明書の有効性は、public-key cryptography の手法で CA の デジタル署名 を確認して検証できます。公開鍵インフラストラクチャー (PKI) 内で信頼されるために、証明書は、PKI に登録されているその他のエンティティーによって信頼されている CA により発行および署名される必要があります。

Certificate Manager

認証局として機能する独立した Certificate System サブシステム。Certificate Manager インスタンスは、証明書を発行、更新、および取り消します。証明書は、CRL とともに LDAP ディレクトリーに公開できます。エンドエンティティーからのリクエストを受け入れます。認証局 (CA) を参照してください。

Certificate Manager エージェント

Certificate Manager のエージェントサービスの管理が許可されているグループに所属するユーザーです。これらのサービスには、証明書要求にアクセスして変更 (承認および拒否) し、証明書を発行する機能が含まれています。

Certificate Request Message Format (CRMF)

X.509 証明書の管理に関連するメッセージに使用される形式。この形式は CMMF のサブセットです。証明書管理メッセージ形式 (CMMF) も参照してください。詳細は、https://tools.ietf.org/html/rfc2511を参照してください。

Certificate System

Red Hat Certificate System、暗号化メッセージ構文 (CS) を参照してください。

Certificate System コンソール

1 つの Certificate System インスタンスで開くことができるコンソール。Certificate System コンソールを使用すると、Certificate System 管理者は、対応する Certificate System インスタンスの設定設定を制御できます。

Certificate System サブシステム

5 つの Certificate System マネージャーの 1 つ: Certificate Manager、Online Certificate Status Manager、キーリカバリー認証局、Token Key Service、または Token Processing System。

chained CA

リンクされた CA を参照してください。

cipher

cryptographic algorithm を参照してください。

CMC

暗号化メッセージ構文 (CMC) を介した証明書管理メッセージ を参照してください。

CMC 登録

署名された登録または署名された失効要求のいずれかを、エージェントの署名証明書を使用して Certificate Manager に送信できるようにする機能。これらの要求は Certificate Manager によって自動的に処理されます。

CMMF

証明書管理メッセージ形式 (CMMF)を参照してください。

CRL

証明書失効リスト (CRL) を参照してください。

CRMF

Certificate Request Message Format (CRMF) を参照してください。

cross-certification

異なる証明書階層またはチェーン内の 2 つの CA による証明書交換クロス認定は、両方の階層に対応できるように信頼チェーンを拡張します。認証局 (CA) も参照してください。

cryptographic algorithm

encryption や decryption などの暗号化操作を実行するために使用される一連のルールまたは方向。

cryptographic module

PKCS #11 モジュール を参照してください。

CSP

暗号化サービスプロバイダー (CSP) を参照してください。

クライアント TLS 証明書

TLS プロトコルを使用してサーバーにクライアントを識別するために使用する証明書。Transport Layer Security (TLS)を参照してください。

クライアント認証

名前とパスワード、または証明書とデジタル署名されたデータなどを使用して、サーバーに対してクライアントを識別するプロセス。証明書ベースの認証、パスワードベースの認証、サーバー認証を参照してください。

ペア間の証明書

ある CA から別の CA に発行され、信頼の輪を形成するために両方の CA によって保存される証明書。2 つの CA は相互に証明書を発行し、両方のクロスペア証明書を証明書ペアとして格納します。

信頼チェーン

証明書チェーン を参照してください。

共通基準

ソフトウェアとハードウェアの両方のコンポーネントについて、コンピューターのセキュリティーを評価する認定基準。ソフトウェアまたはハードウェアのベンダーは、動作環境と指定された設定を定義し、脅威を特定し、評価対象 (評価対象) の開発プロセスと展開プロセスの両方を概説します。次に、Common Criteria 認定ラボは、実装設計をテストして脆弱性を探します。

暗号化サービスプロバイダー (CSP)

PKCS #11 で定義されているような標準インターフェイスを使用してそのようなサービスを要求するソフトウェアに代わって、キー生成、キーストレージ、暗号化などの暗号化サービスを実行する暗号化モジュール。

暗号化メッセージ構文 (CMC) を介した証明書管理メッセージ

Certificate Manager への証明書の要求を伝えるために使用するメッセージ形式。Internet Engineering Task Force (IETF) PKIX の作業グループから提案された標準。詳細は、https://tools.ietf.org/html/draft-ietf-pkix-cmc-02 を参照してください。

暗号化メッセージ構文 (CS)

CMMF などの任意のメッセージにデジタル署名、ダイジェスト、認証、または暗号化するために使用される構文。

証明書のフィンガープリント

証明書に関連付けられた one-way hash。番号は証明書自体の一部ではありませんが、証明書の内容にハッシュ関数を適用することによって生成されます。証明書の内容が 1 文字でも変更されると、同じ関数で異なる番号が生成されます。したがって、証明書のフィンガープリントを使用して、証明書が改ざんされていないことを確認できます。

証明書の拡張

X.509 v3 証明書には、証明書に任意の数の追加フィールドを追加できる拡張フィールドが含まれています。証明書拡張機能は、代替サブジェクト名や使用制限などの情報を証明書に追加する方法を提供します。PKIX ワーキンググループによって、いくつかの標準拡張機能が定義されています。

証明書チェーン

連続した認証局によって署名された証明書の階層セット。CA 証明書は 認証局 (CA) を識別し、その認証局が発行した証明書の署名に使用されます。CA 証明書は、親 CA の CA 証明書により署名され、root CA まで署名できます。Certificate System により、エンドエンティティーが証明書チェーンのすべての証明書を取得できるようになります。

証明書プロファイル

特定のタイプの登録を定義する一連の設定設定。証明書プロファイルは、証明書プロファイルの認証方法とともに、特定のタイプの登録のポリシーを設定します。

証明書ベースの認証

証明書および公開鍵暗号に基づく認証。パスワードベースの認証 も参照してください。

証明書失効リスト (CRL)

X.509 標準で定義されているように、認証局 (CA) により生成され署名されたシリアル番号ごとに取り消された証明書のリスト。

証明書管理メッセージ形式 (CMMF)

エンドエンティティーから Certificate Manager に証明書要求と失効要求を伝達し、エンドエンティティーにさまざまな情報を送信するために使用されるメッセージ形式。Internet Engineering Task Force (IETF) PKIX の作業グループから提案された標準。CMMF は、別の提案された標準 暗号化メッセージ構文 (CMC) を介した証明書管理メッセージ に含まれています。詳細は、https://tools.ietf.org/html/draft-ietf-pkix-cmmf-02 を参照してください。

認証局 (CA)

証明書が特定を意図している個人またはエンティティーの身元を検証した後、certificate を発行する信頼されたエンティティー。CA は証明書を更新し、取り消し、CRL を生成します。証明書の発行者フィールドで指定されたエンティティーは、常に CA です。認証局は、独立したサードパーティー、または Red Hat Certificate System などの証明書発行サーバーソフトウェアを使用する個人または組織にすることができます。

D

decryption

暗号化されたデータのスクランブル解除。encryption を参照してください。

digital ID

certificate を参照してください。

キーリカバリー認証局

エンドエンティティーの RSA 暗号化キーの長期アーカイブとリカバリーを管理する任意の独立した Certificate System サブシステム。Certificate Manager は、新しい証明書を発行する前に、キーリカバリー機関を使用してエンドエンティティーの暗号化キーをアーカイブするように設定できます。キーリカバリー機関は、エンドエンティティーが機密性の高い電子メールなど、組織がいつかリカバリーする必要のあるデータを暗号化している場合にのみ役立ちます。デュアルキーペアをサポートするエンドエンティティーでのみ使用できます。2 つの個別のキーペアで、1 つは暗号化用、もう 1 つはデジタル署名用です。

キーリカバリー認証局のストレージキー

キーリカバリー機関の秘密トランスポートキーで復号された後、エンドエンティティーの暗号化キーを暗号化するためにキーリカバリー機関によって使用される特別なキー。ストレージキーがキーリカバリー機関を離れることはありません。

キーリカバリー認証局のトランスポート証明書

エンドエンティティーがキーリカバリー機関に転送するためにエンティティーの暗号化キーを暗号化するために使用する公開キーを認証します。キーリカバリー機関は、認証された公開鍵に対応する秘密鍵を使用して、ストレージ鍵で暗号化する前に、エンドエンティティーの鍵を復号します。

キーリカバリー認証局エージェント

要求キューの管理や HTML ベースの管理ページを使用したリカバリー操作の許可など、キーリカバリー機関のエージェントサービスの管理を許可されたグループに属するユーザー。

キーリカバリー認証局リカバリーエージェント

キーリカバリー認証局 のストレージキーの一部を所有している **m of n** 人の 1 人。

デジタル署名

デジタル署名を作成するため、署名ソフトウェアは最初に、新しく発行された証明書など、署名するデータから one-way hash を作成します。次に、一方向ハッシュは署名者の秘密鍵で暗号化されます。作成されるデジタル署名は、署名されるデータごとに一意になります。1 つのコンマがメッセージに追加されていても、そのメッセージのデジタル署名が変更されます。署名者の公開鍵を使用したデジタル署名の復号に成功し、同じデータの別のハッシュと比較することで、改ざんの検出 が可能になります。公開鍵を含む証明書の 証明書チェーン の検証により、署名側の認証が提供されます。nonrepudiation、encryptionも参照してください。

デュアルキーペア

2 つの個別の証明書に対応する、2 つの公開鍵と秘密鍵のペア (合計 4 つの鍵)。一方のペアの秘密鍵は署名操作に使用され、もう一方のペアの公開鍵と秘密鍵は暗号化および復号操作に使用されます。各ペアは個別の certificate に対応します。暗号化、public-key cryptography、署名鍵 も参照してください。

デルタ CRL

最後の完全な CRL が発行されてから取り消された証明書の一覧を含む CRL。

分散ポイント

証明書セットを定義するのに CRL に使用されます。各ディストリビューションポイントは、発行する証明書のセットにより定義されます。CRL は、特定のディストリビューションポイント用に作成できます。

識別名 (DN)

証明書の件名を特定する一連の AVA。属性値アサーション (AVA) を参照してください。

E

eavesdropping

情報が意図されていないエンティティーによってネットワークを介して送信された情報の不正な傍受。

encryption

その意味を偽装する方法で情報をスクランブルします。decryption を参照してください。

enrollment

公開鍵インフラストラクチャー (PKI) で使用する X.509 証明書を要求および受信するプロセス。**登録** としても知られています。

extensions フィールド

証明書の拡張 を参照してください。

エンドエンティティー

公開鍵インフラストラクチャー (PKI)、人、ルーター、サーバー、またはその他のエンティティーで、certificate を使用してそれ自体を特定します。

暗号化

暗号化のみに使用される秘密鍵。暗号化鍵とその同等の公開鍵、および 署名鍵 とその同等の公開鍵は、デュアルキーペア を設定します。

楕円曲線暗号 (ECC)

暗号鍵の基礎となる数学的な問題に対して、楕円曲線を用いて加算対数を作成する暗号アルゴリズム。ECC 暗号は、RSA 暗号よりも効率的に使用でき、本質的に複雑であるため、RSA 暗号よりも小さいビットで強力です。

F

Federal Bridge Certificate Authority (FBCA)

2 つの CA が相互にクロスペア証明書を発行し、2 つのクロスペア証明書を単一の証明書ペアとして格納することにより、信頼の輪を形成する設定。

fingerprint

証明書のフィンガープリント を参照してください。

FIPS PUBS 140

FIPS PUBS (Federal Information Standards Publications) 140 は、データの暗号化と復号、またはデジタル署名の作成や検証などの他の暗号化操作を実行する暗号化モジュール、ハードウェア、またはソフトウェアの実装に関する米国政府の標準です。米国政府に販売される多くの製品は、1 つ以上の FIPS 標準に準拠する必要があります。http://www.nist.gov/itl/fipscurrent.cfm を参照してください。

firewall

2 つ以上のネットワーク間の境界を強制するシステムまたはシステムの組み合わせ。

I

impersonation

ネットワークを介して送信される情報の意図された受信者を装う行為。なりすましには、spoofing と 詐称 の 2 つの形式があります。

input

証明書プロファイル機能のコンテキストでは、特定の証明書プロファイルの登録フォームを定義します。各入力が設定され、この登録用に設定されたすべての入力から登録フォームが動的に作成されます。

intermediate CA

ルート CA と 証明書チェーン で発行した証明書との間の証明書のある CA。

IP スプーフィング

クライアント IP アドレスの禁止。

J

JAR ファイル

Java™ アーカイブ (JAR) 形式 に従って編成されたファイルの圧縮コレクションのデジタルエンベロープ。

Java™ アーカイブ (JAR) 形式

デジタル署名、インストーラスクリプト、およびその他の情報をディレクトリー内のファイルに関連付けるための一連の規則。

Java™ セキュリティーサービス (JSS)

あJava™ネットワークセキュリティーサービス (NSS) によって実行されるセキュリティー操作を制御するためのインターフェイス。

Java™ ネイティブインターフェイス (JNI)

特定のプラットフォーム上の Java™ 仮想マシン (JVM) の異なる実装間でバイナリー互換性を提供する標準的なプログラミングインターフェイスで、単一のプラットフォーム用に C や C++ などの言語で記述された既存のコードを Java™ にバインドできるようにします。http://java.sun.com/products/jdk/1.2/docs/guide/jni/index.html を参照してください。

Java™ 暗号アーキテクチャー (JCA)

暗号化サービス用に Sun Microsystems によって開発された API 仕様および参照。http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.Introduction を参照してください。

Java™ 開発キット (JDK)

Java™ プログラミング言語を使用してアプリケーションとアプレットを開発するために Sun Microsystems が提供するソフトウェア開発キット。

K

KEA

鍵交換アルゴリズム (KEA) を参照してください。

key

データを暗号化または復号化するために、cryptographic algorithm が使用する多数の数値。たとえば、あるユーザーの 公開鍵 にあるユーザーは、そのユーザー専用のメッセージを暗号化できます。その後、メッセージは対応する プライベートキー を使用して復号化する必要があります。

key exchange

TLS セッション中に両方が使用する対称鍵を決定するために、クライアントとサーバーが従う手順。

鍵交換アルゴリズム (KEA)

米国政府が鍵交換に使用するアルゴリズム。

L

Lightweight Directory Access Protocol (LDAP)

TCP/IP および複数のプラットフォームで実行されるためのディレクトリーサービスプロトコル。LDAP は、X.500 ディレクトリーへのアクセスに使用される Directory Access Protocol (DAP) の簡易バージョンです。LDAP は IETF の変更制御下にあり、インターネット要件を満たすために進化しています。

リンクされた CA

公開サードパーティーの CA によって署名される証明書が内部でデプロイされる 認証局 (CA)。内部 CA は、発行する証明書のルート CA として機能し、サードパーティー CA は、同じサードパーティールート CA にリンクされている他の CA によって発行された証明書のルート CA として機能します。チェーン CA としても知られており、異なるパブリック CA で使用される他の用語も使用します。

M

MD5

Ronald Rivest によって開発されたメッセージダイジェストアルゴリズム。one-way hash も参照してください。

message digest

one-way hash を参照してください。

手動認証

各証明書要求の人間による承認を必要とする Certificate System サブシステムを設定する方法。この形式の認証では、サーブレットは、認証モジュールの処理が成功した後、証明書要求を要求キューに転送します。次に、適切な権限を持つエージェントは、プロファイルの処理と証明書の発行を続行する前に、各要求を個別に承認する必要があります。

詐称

そうではない個人または組織としてのエンティティーの提示。たとえば、Web サイトが実際にはクレジットカードでの支払いを行いますが、商品を送信しないサイトである場合、その Web サイトは家具店のふりをする可能性があります。虚偽表示は impersonation の 1 つの形式です。spoofing も参照してください。

N

non-TMS

トークン以外の管理システム。スマートカードを直接処理 **しない** サブシステム (CA、および任意で KRA と OCSP) の設定を指します。

トークン管理システム参照

nonrepudiation

メッセージの送信を拒否するためのメッセージの送信者による信頼性。デジタル署名 は、否認防止の形式を 1 つ提供します。

ネットワークセキュリティーサービス (NSS)

セキュリティー対応の通信アプリケーションのクロスプラットフォーム開発をサポートするように設計されたライブラリーのセット。NSS ライブラリーを使用して構築されたアプリケーションは、認証、改ざん検出、および暗号化のための Transport Layer Security (TLS) プロトコル、および暗号化トークンインターフェイスのための PKCS #11 プロトコルをサポートします。NSS は、ソフトウェア開発キットとして個別に利用できます。

O

OCSP

オンライン証明書ステータスプロトコル

one-way hash

1.ハッシュアルゴリズムを使用して任意の長さのデータから生成された固定長の数。メッセージダイジェストとも呼ばれる数字は、ハッシュされたデータに固有のものです。1 文字を削除または変更しても、データの変更は異なります。

2.ハッシュされたデータの内容をハッシュから推測することはできません。

operation

アクセス制御命令で許可または拒否されている、読み取りや書き込みなどの特定の操作。

output

証明書プロファイル機能のコンテキストでは、特定の証明書プロファイルの証明書登録が成功した結果のフォームを定義します。各出力が設定され、この登録に設定されたすべての出力からフォームを動的に作成します。

オブジェクトの署名

ソフトウェア開発者が Java コード、JavaScript スクリプト、またはあらゆる種類のファイルに署名し、ユーザーが署名者を識別し、署名されたコードによってローカルシステムリソースへのアクセスを制御できるようにするファイル署名の方法。

オブジェクト署名証明書

オブジェクトの署名 に関連する秘密鍵がオブジェクトの署名に使用される証明書。

P

PKCS #10

証明書要求を制御する公開鍵暗号標準規格。

PKCS #11

スマートカードなどの暗号化トークンを管理する公開鍵暗号化標準。

PKCS #11 モジュール

PKCS #11 インターフェイスを介して、暗号化サービス (暗号化や復号など) を提供する暗号化デバイスのドライバー。**暗号化モジュール** または **暗号化サービスプロバイダー** とも呼ばれる PKCS #11 モジュールは、ハードウェアまたはソフトウェアのいずれかで実装できます。PKCS #11 モジュールには、常に 1 つ以上のスロットがあり、スマートカードなどの物理リーダーの形式で物理ハードウェアスロットとして、またはソフトウェアの概念スロットとして実装できます。PKCS #11 モジュールの各スロットには、トークンを追加できます。このトークンは、暗号化サービスを実際に提供し、必要に応じて証明書と鍵を保存するハードウェアまたはソフトウェアのデバイスです。Red Hat は、Certificate System とともに、ビルトイン PKCS #11 モジュールを提供します。

PKCS #12

鍵の移植性を管理する公開鍵暗号化標準。

PKCS #7

署名と暗号化を制御する公開鍵暗号標準。

POA (proof-of-archival)

キーのシリアル番号、キーリカバリー機関の名前、対応する証明書の subject name、およびアーカイブの日付など、アーカイブされたエンドエンティティーキーに関する情報を含む秘密キーリカバリー機関のトランスポートキーで署名されたデータ。署名されたアーカイブ証明データは、キーのアーカイブ操作が成功した後、キーリカバリー機関から Certificate Manager に返される応答です。キーリカバリー認証局のトランスポート証明書も参照してください。

public-key cryptography

エンティティーがその ID を電子的に検証したり、電子データに署名して暗号化したりできるようにする、確立された技術と標準のセット。公開鍵と秘密鍵の 2 つの鍵が関係します。公開鍵 は、特定の ID にキーを関連付ける証明書の一部として公開されます。対応する秘密鍵はシークレットに保存されます。公開鍵で暗号化したデータは、秘密鍵でのみ復号できます。

パスワードベースの認証

名前とパスワードによる確実な識別。認証、証明書ベースの認証 も参照してください。

プライベートキー

公開鍵暗号で使用されるキーペアの 1 つ。秘密鍵は秘密を保持し、対応する 公開鍵 で暗号化されたデータの復号に使用されます。

公開鍵

公開鍵暗号で使用されるキーペアの 1 つ。公開鍵は自由に配布され、certificate の一部として公開されます。これは通常、公開鍵の所有者に送信されるデータを暗号化するために使用され、所有者は対応する プライベートキー でデータを復号化します。

公開鍵インフラストラクチャー (PKI)

ネットワーク環境での公開鍵暗号化と X.509 v3 証明書の使用を容易にする標準とサービス。

R

RC2, RC4

Rivest による RSA データセキュリティー向けに開発された暗号化アルゴリズム。cryptographic algorithm も併せて参照してください。

Red Hat Certificate System

証明書を作成、デプロイ、および管理するための高度に設定可能なソフトウェアコンポーネントとツールのセット。Certificate System は、さまざまな物理的な場所にあるさまざまな Certificate System インスタンスにインストールできる 5 つの主要なサブシステム (Certificate Manager、Online Certificate Status Manager、キーリカバリー認証局、Token Key Service, and Token Processing System) で設定されています。

root CA

証明書チェーンの上部に自己署名証明書が含まれている 認証局 (CA)。CA 証明書、subordinate CA も参照してください。

RSA algorithm

Rivest-Shamir-Adleman の略で、暗号化と認証の両方の公開鍵アルゴリズムです。Ronald Rivest、Adi Shamir、および Leonard Adleman により開発され、1978 で導入されました。

RSA キー交換

RSA アルゴリズムに基づく TLS の鍵交換アルゴリズム。

登録

enrollment を参照してください。

S

sandbox

Java™ コードが動作する必要がある、注意して定義された制限の Java™ 用語。

servlet

Certificate System サブシステムに代わってエンドエンティティーとの特定の種類の相互作用を処理する Java™ コード。たとえば、証明書の登録、失効、およびキーリカバリー要求は、それぞれ別のサーブレットで処理されます。

SHA-1

セキュアなハッシュアルゴリズム (米国政府が使用するハッシュ関数)。

signature algorithm

デジタル署名の作成に使用される暗号化アルゴリズム。Certificate System は、MD5 および SHA-1 署名アルゴリズムに対応します。cryptographic algorithm、デジタル署名 も併せて参照してください。

slot

PKCS #11 モジュール の一部 (ハードウェアまたはソフトウェアのいずれか)。これには token が含まれています。

spoofing

他人のふりをします。たとえば、あるユーザーがメールアドレス **jdoe@example.com** やコンピューターから、**www.redhat.com** と呼ばれるサイトとして誤って特定できます。なりすましは、impersonation の 1 つの形です。詐称も併せて参照してください。

subject

certificate で識別されるエンティティー。特に、証明書のサブジェクトフィールドには、認定されたエンティティーを一意に識別する subject name が含まれます。

subject name

certificate の subject を個別に記述する 識別名 (DN)。

subordinate CA

証明書が別の下位 CA またはルート CA によって署名されている認証局。CA 証明書、root CAを参照してください。

symmetric encryption

同じ暗号化キーを使用して特定のメッセージを暗号化および復号する暗号化方式。

TLS

Transport Layer Security (TLS) を参照してください。

サーバー TLS 証明書

Transport Layer Security (TLS) プロトコルを使用して、クライアントにサーバーを識別するために使用する証明書。

サーバー認証

クライアントにサーバーを特定するプロセス。クライアント認証も併せて参照してください。

シングルサインオン

1.Certificate System において、内部データベースとトークンのパスワードを保管することにより、Red Hat Certificate System へのサインオン方法を簡素化するパスワード。ユーザーがログインするたびに、この単一のパスワードを入力する必要があります。

2.ユーザーが 1 台のコンピューターに一度ログインし、ネットワーク内のさまざまなサーバーによって自動的に認証される機能。部分的なシングルサインオンソリューションは、さまざまなサーバーで使用されるパスワードを自動的に追跡するメカニズムなど、さまざまな形式をとることができます。証明書は、公開鍵インフラストラクチャー (PKI) 内のシングルサインオンをサポートします。ユーザーは、ローカルクライアントの秘密鍵データベースに一度ログインし、クライアントソフトウェアが動作している限り、証明書ベースの認証 を使用して、ユーザーがアクセスを許可されている組織内の各サーバーにアクセスすることができます。

スマートカード

マイクロプロセッサーを搭載し、キーや証明書などの暗号化情報を格納し、暗号化操作を実行する小さなデバイス。スマートカードは、一部またはすべて PKCS #11 インターフェイスを実装します。

セキュアなチャンネル

TPS とスマートカード間のセキュリティーアソシエーション。TKS とスマートカード APDU によって生成された共有マスターキーに基づいて暗号化された通信を可能にします。

セキュリティードメイン

PKI サブシステムの集中リポジトリーまたはインベントリー。その主な目的は、サブシステム間の信頼できる関係を自動的に確立することにより、新しい PKI サービスのインストールと設定を容易にすることです。

セルフテスト

インスタンスの起動時とオンデマンドの両方の Certificate System インスタンスをテストする機能。

署名付き監査ログ

監査ログ を参照してください。

署名証明書

公開鍵がデジタル署名の作成に使用される秘密鍵に対応する証明書。たとえば、Certificate Manager には、発行する証明書の署名に使用する秘密鍵に対応する公開鍵を持つ署名証明書が必要です。

署名鍵

署名用途に使用する秘密鍵署名鍵とその同等の公開鍵、および 暗号化 とその同等の公開鍵は、デュアルキーペア を設定します。

T

token

PKCS #11 モジュール の slot に関連付けられたハードウェアまたはソフトウェアのデバイス暗号化サービスを提供し、必要に応じて証明書および鍵を保存します。

Transport Layer Security (TLS)

クライアントとサーバーとの間の相互認証と、認証および暗号化された接続の確立を可能にするプロトコル。TLS は、TCP/IP の上で実行され、HTTP、LDAP、IMAP、NNTP、およびその他の高レベルのネットワークプロトコルの下で実行されます。

trust

個人または他のエンティティーに確定します。公開鍵インフラストラクチャー (PKI) では、信頼とは、証明書のユーザーと、その証明書を発行した 認証局 (CA) との間の関係を指します。CA が信頼されている場合は、その CA が発行する有効な証明書を信頼できます。

ツリー階層

LDAP ディレクトリーの階層構造。

トークンキーサービス (TKS)

スマートカードの APDU およびトークン CUID などの他の共有情報に基づいて、スマートカードごとに特定の個別のキーを取得するトークン管理システムのサブシステム。

トークン処理システム (TPS)

Enterprise Security Client とスマートカードを直接対話して、これらのスマートカードのキーと証明書を管理するサブシステム。

トークン管理システム (TMS)

相互に関連するサブシステム (CA、TKS、TPS、および任意で KRA) は、スマートカード (トークン) の証明書を管理するために使用されます。

改ざんの検出

電子形式で受信したデータが同じデータの元のバージョンと完全に対応することを保証するメカニズム。

V

仮想プライベートネットワーク (VPN)

企業の地理的に離れた部署を接続する方法。VPN を使用すると、部署間は暗号化されたチャンネルを介して通信できるため、通常はプライベートネットワークに制限される認証済みの機密トランザクションが可能になります。

索引

シンボル

A

B

C

付録F 改訂履歴

改訂番号は本ガイドに関するものであり、Red Hat Certificate System のバージョン番号とは関係ありません。

改訂 9.4-1 **Thu Feb 11, 2021**
9.4 Common Criteria Maintenance Update のさまざまなマイナー修正。

改訂 9.4-0 **Wed Apr 10, 2019**
本ガイドの初期バージョン