Red Hat Certificate System Red Hat Certificate System 9

コマンドラインツールガイド

リファレンスガイド

Last Updated: 2023-05-10

Red Hat Certificate System Red Hat Certificate System 9 コマンドラインツールガイド

リファレンスガイド

Petr Bokoč
Red Hat Customer Content Services
pbokoc@redhat.com

Tomáš Čapek
Red Hat Customer Content Services

Aneta Petrová
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律上の通知

概要

本ガイドでは、サブシステムインスタンスの作成、削除、管理、および鍵と証明書の作成と管理に使用できる重要なコマンドラインユーティリティーを説明します。

目次

第1章 PKISPAWN ユーティリティーおよび PKIDESTROY ユーティリティー

Certificate System には、サブシステムを作成および削除する 2 つのコマンドラインユーティリティー(**pkispawn** および **pkidestroy**)が含まれています。

注記

以前のバージョンの Certificate System では、インストールおよび設定は、**pkicreate** ユーティリティーおよび **pkisilent** ユーティリティーが管理する 2 つの別個のタスクに分割されていました。Certificate System バージョン 9 以降では、単一の **pkispawn** ユーティリティーが、これらすべての操作を管理するようになりました。

pkiremove ユーティリティーを使用して、以前の Certificate System バージョンのサブシステムが削除されました。ユーティリティーが **pkidestroy** に置き換えられました。

1.1. PKISPAWN ユーティリティー

pkispawn ユーティリティーは、Certificate System サブシステムを作成し、設定します。以下の 2 つのインストールモードをサポートしています。

- 非対話モード。ユーザーは、コマンドラインオプションと設定ファイルを使用してインストールと設定設定を提供します。
- インタラクティブモード。**pkispawn** は、インストールに必要な基本情報の入力を自動的に要求します。

両方のインストールモードを組み合わせることもできます。これにより、一部の設定を **pkispawn** に直接渡すことができ、ユーティリティープロンプトに他の設定を対話的に行うことができます。たとえば、**-s** オプションを **pkispawn** に追加し、設定ファイルを提供するために **-f** オプションを指定しないと、インストールは **/etc/pki/default.cfg** ファイルからデフォルト設定を使用し、パスワードなどの追加のカスタム情報を対話的に要求します。

本セクションでは、**pkispawn** を使用して Certificate System サブシステムをインストールする方法を説明します。**pkispawn** の詳細は、pkispawn(8) の man ページを参照してください。man ページには、**pkispawn** の使用方法に関するさまざまな例が記載されています。

1.1.1. 非対話の pkispawn モード

設定パラメーターは事前に作成された設定ファイルで提供されるため、ユーティリティーはいくつかのコマンドラインオプションのみを受け入れます。

pkispawn を使用してサブシステムを作成および設定するには、以下のオプションを指定して ユーティリティーを実行します。

-s オプション

作成して設定するサブシステムを指定します(CA、KRA、OCSP、TKS、または TPS)。

-f オプション

設定ファイルへのパスを指定します。

たとえば、以下のコマンドは **myconfig.txt** ファイルに基づいて CA サブシステムを作成します。

```
# pkispawn -s CA -f myconfig.txt
```

pkispawnの設定ファイル

Certificate System は、**/etc/pki/default.cfg** ファイルにデフォルト設定を保存します。**pkispawn** ユーティリティーに提供できるカスタム設定ファイルを作成するには、**default.cfg** を別の場所にコピーします。次に、コピーしたファイルを変更して、**pkispawn** が新しいサブシステムに適用する設定を定義します。

カスタム設定ファイルは、デフォルトの **default.cfg** ファイルよりも優先されます。一般的には、ユーザーによって提供されるカスタム設定ファイルにデフォルト設定とは異なるパラメーターのみを保存するのが一般的です。

default.cfg ファイルは複数のセクションに分かれています。

一般的なセクション

一般的なセクションには、デフォルトの設定オプションおよび Tomcat 設定オプションが含まれます。以下に例を示します。

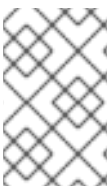
```
[DEFAULT]
pki_admin_password=
pki_backup_password=
pki_client_database_password=
pki_client_pkcs12_password=
pki_ds_password=
pki_replication_password=
pki_security_domain_password=
pki_token_password=
[Tomcat]
pki_clone_pkcs12_password=
```

サブシステム固有のセクション

サブシステムセクションには、サブシステム固有の設定オプションが含まれています。以下に例を示します。

```
[CA]
pki_admin_name=caadmin
pki_admin_email=caadmin@example.com
```

後のセクションで定義された設定は、前のセクションの設定よりも優先されます。たとえば、サブシステム固有のセクションの設定は、**Tomcat** セクションよりも優先されます。これは、**DEFAULT** セクションの設定よりも優先されます。この動作により、**DEFAULT** セクションまたは **Tomcat** セクションのすべてのサブシステムで共有されるパラメーターと、そのサブシステムの セクションで特定のサブシステムに固有のオプションを指定できます。



注記

default.cfg ファイルのコピーは、**pkispawn** の実行後に作成されたサブシステム内に保存されます。その後、**pkidestroy** でサブシステムを削除するときにコピーが使用されます。

pkispawn に提供できるさまざまなカスタム設定ファイルの例は、pkispawn(8) の man ページを参照してください。**default.cfg** の詳細は、pki_default.cfg(5) の man ページを参照してください。

1.1.2. インタラクティブな pkispawn モード

pkispawn に設定オプションを指定しないと、ユーティリティーは対話式インストールモードに入り、基本的な必要なインストールオプションを自動的に要求します。インタラクティブな **pkispawn** インストールモードは、Certificate System に精通しているユーザーに適しています。インタラクティブモードに使用する基本的なオプションの一覧は、pkispawn(8) の man ページを参照してください。

対話型インストールでは、他の設定オプションは利用できません。高度な設定を使用する場合は、「非対話の **pkispawn** モード」 で説明されているように、**-f** オプションを使用して **pkispawn** に設定ファイルを指定します。

対話型インストールモードで指定されたパラメーターは、**/etc/sysconfig/pki/tomcat/*instance_name*/*サブシステム*/deployment.cfg** ファイルに保存されます。

1.1.3. 単一インスタンスでの複数のサブシステムの作成

単一の Tomcat インスタンスには複数のサブシステムを含めることができます。ただし、1 つのインスタンスに含めることができるサブシステムのタイプは 1 つだけです。たとえば、インスタンスには 1 つの CA と 1 つの KRA サブシステムを含めることができますが、2 つの CA または 2 つの KRA サブシステムを含めることはできません。

複数のサブシステムでインスタンスを作成するには、**pkispawn** を複数回実行し、毎回異なるサブシステムを指定します。たとえば、CA および KRA を使用してインスタンスを作成するには、**pkispawn -s CA** コマンドを実行し、**pkispawn -s KRA** コマンドを実行します。

1.1.4. 共有および非共有インスタンス

共有 PKI インスタンスでは、共有インスタンス内のすべてのサブシステムが同じインスタンス名とポートを使用します。*共有されていない* PKI インスタンスでは、インスタンスが別の PKI インスタンスと同じマシンに存在する場合に、サブシステムは一意のインスタンス名およびポートを使用します。このような PKI インスタンスをインストールする場合は、**pkispawn** 設定ファイルに必要なパラメーターを定義します。

注記

インスタンスは、インストールされるサブシステムのタイプに関係なく、共有されていないインスタンスとして常に作成されます。別のタイプの 2 つ目のサブシステムをインストールすると、インスタンスが共有インスタンスになります。

CA がインストールされているマシンとは異なるマシンに共有 Tomcat インスタンスをインストールするには、**KRA、OCSP、TKS、または TPS 設定ファイルに** 必要なパラメーターについて、pkispawn(1) の man ページの KRA、OCSP、または TKS の例を参照してください。PKI インスタンスのカスタム名を指定する場合は、ファイルの **DEFAULT** セクションに ***pki_instance_name*** パラメーターも定義します。

CA がインストールされている同じマシンに非共有 Tomcat インスタンスをインストールするには、上記の例に従って、一意のインスタンス名およびポートも定義します。これを行うには、KRA、OCSP、TKS、または TPS 設定ファイルで以下のパラメーターを使用します。

```
[DEFAULT]
pki_instance_name=unique_value
pki_http_port=unique_value
pki_https_port=unique_value
```

```
[Tomcat]
pki_ajp_port=unique_value
pki_tomcat_server=unique_value
```

1.2. PKIDESTROY ユーティリティー

pkidestroy ユーティリティーは、指定された Certificate Server インスタンスからサブシステムを削除します。このユーティリティーは非対話的に実行することも、対話的に実行することもできます。

1.2.1. 非対話型 pkidestroy モード

pkidestroy を使用してサブシステムを削除するには、以下のオプションを指定して ユーティリティー を実行します。

-s オプション

削除するサブシステムを指定します(CA、KRA、OCSP、TKS、または TPS)。

-i オプション

サブシステムを削除するインスタンスの名前を指定します。

たとえば、以下のコマンドは *instance_name* という名前のインスタンスから KRA サブシステムを削除します。

```
# pkidestroy -s KRA -i instance_name
```

pkidestroy の詳細は、pkidestroy(8) の man ページを参照してください。

1.2.2. インタラクティブな pkidestroy モード

オプションを指定せずに **pkidestroy** を実行すると、ユーティリティーは必要な情報を自動的に要求します。たとえば、**-s** オプションを指定しないと、**pkispawn** はサブシステムを削除するように対話的に要求します。

第2章 PKI ユーティリティー

pki ユーティリティーを使用すると、クライアントは Certificate System サーバーの PKI サービスにアクセスできます。ユーティリティーは、ユーザーまたはグループの管理、証明書管理、プロファイル管理など、さまざまな操作を実行するために設計されたコマンドおよびサブコマンドを多数提供します。

利用可能な **pki** コマンドおよびオプションをすべて表示するには、引数を指定せずに **pki** を実行します。

```
$ pki

usage: pki [OPTIONS..] <command> [ARGS..]
 -c <password>                Security database password
 -d <database>                Security database location (default:
      ~/.dogtag/nssdb)
...

Subsystems:
 ca   CA management commands
 kra   KRA management commands
 ocsp   OCSP management commands
...

Commands:
 client               Client management commands
 cert                 Certificate management commands
 group                 Group management commands
...
```

pki コマンドにはサブコマンドがあります。特定の **pki** コマンドで使用できるサブコマンドを表示するには、オプションを指定せずにコマンドを実行します。たとえば、**pki クライアント** コマンドで利用可能なサブコマンドを表示するには、次のコマンドを実行します。

```
$ pki client

Commands:
 client-init            Initialize client security database
 client-cert-find       Find certificates in client security database
 client-cert-import     Import certificate into client security database
...
```

2.1. 接続パラメーター

pki ユーティリティーは、デフォルトで以下のパラメーターで PKI サーバーに接続します。

- プロトコル： **http**
- ホスト名： **localhost**
- ポート： **8080**

pki コマンドに以下のオプションを追加すると、カスタムパラメーターを手動で指定できます。

- **-p** は、プロトコルを指定します。

- **-h** はホスト名を指定します。
- **-p**: ポートを指定します。

以下に例を示します。

```
pki -P https -h server.example.com -p 8443 cert-find
```

接続パラメーターを URL として指定することもできます。これを行うには、**-U** オプションを使用して *protocol://hostname:port* 形式の URL を指定します。サブシステムは、実行中のコマンドに基づいて決定されます。たとえば、次のコマンドは CA の証明書を一覧表示します。

```
pki -U https://server.example.com:8443 cert-find
```

2.2. 認証

pki ユーティリティーに基づいたコマンドによっては、ユーザーが認証する必要があります。ユーティリティーは、ユーザー名とパスワードの認証情報、またはクライアント証明書を使用した認証をサポートします。

ユーザー名とパスワードによる認証

ユーザー名を指定するには、特定の **pki** コマンドに **-u** オプションを追加します。パスワードを指定するには、 **-W** オプションまたは **-w** オプションを使用します。または、-W または -w を使用してコマンドに直接パスワードを追加しない場合、**pki** は必要に応じてパスワードを対話的に要求します。

バッチ操作 の場合は、**-W** を使用してパスワードを指定することが推奨されます。このオプションを使用すると、システムパーミッション、システム ACL、SELinux ポリシーの設定など、パスワードを保護するための特定のセキュリティー対策を取ることができます。**-w** を使用すると、パスワードをプレーンテキストで指定します。

個々のコマンドラインの呼び出し では、コマンドでパスワードを直接指定せず、代わりに対話的に指定することが推奨されます。たとえば、以下のコマンドを実行して、ユーザーはユーザー名のみを提供し、**pki** プロンプトにパスワードを指定します。

```
pki -u user_name user-find
```

説明されているオプションの詳細は、pki(1) の man ページを参照してください。

クライアント証明書を使用した認証

必要な証明書情報を指定するには、**-C** オプションまたは **-c** オプションを使用してセキュリティーデータベースファイルを指定し、**-n** オプションを使用して証明書のニックネームを指定します。

バッチ操作 の場合は、**-C** を使用してファイルを渡すことが推奨されます。このオプションを使用すると、システムパーミッション、システム ACL、SELinux ポリシーの設定など、ファイルを保護するための特定のセキュリティー対策を取ることができます。**-c** では、ファイルはプレーンテキストで提供されます。

```
pki -C security_database_password_file -n certificate_nickname user-find
```

説明されているオプションの詳細は、pki(1) の man ページを参照してください。

2.3. ページング PKI コマンドの出力

pki ユーティリティーはページネーションをサポートします。コマンドの出力を複数のページに分割し、指定したページを 1 つだけ表示できます。ページネーションは、**cert-find** コマンドなど、多くの結果を表示するコマンドに特に便利です。

pki コマンドの出力をページに分割するには、コマンドを入力するときに以下のオプションを使用します。

- **--start** は、ページの最初のエントリーのインデックスを定義します。コマンド出力の最初のエントリーで開始する場合は、このオプションを **0**に設定します。
- **--size** は、ページ内のエントリー数を定義します。

たとえば、10 エントリーで **pki user-find** コマンドの最初のページを要求するには、次のコマンドを実行します。

```
$ pki user-find --start 0 --size 10
```

出力の 2 ページ目を要求するには、以下を実行します。

```
$ pki user-find --start 10 -- size 10
```

2.4. サポートされている PKI コマンドの概要

本セクションでは、**pki** コマンドとそのサブコマンドの一部と、その機能を紹介します。特定の **pki** サブコマンドの使用方法に関する詳細情報は、**--help** オプションを追加して実行します。以下に例を示します。

```
$ pki cert-find --help
usage: cert-find [OPTIONS...]
--certTypeSecureEmail <on|off>        Certifiate Type: Secure Email
--certTypeSSLClient <on|off>          Certifiate Type: SSL Client
--certTypeSSLServer <on|off>          Certifiate Type: SSL Server
...
```

サブコマンドの一部は、pki(1) の man ページにも記載されています。

2.4.1. pki クライアントによるクライアント管理

pki client-* コマンドを使用すると、Certificate System クライアント環境を管理できます。これらのコマンドの詳細は、pki-client(1) の man ページを参照してください。

クライアントの初期化

pki client-init

新しいクライアント環境を初期化します。このコマンドは、デフォルトの証明書データベースディレクトリー ~/**.dogtag**/**nssdb**/ にセキュリティーデータベースを作成します。新しいセキュリティーデータベースのパスワードは、**-c** オプションまたは **-C** オプションで指定する必要があります。以下に例を示します。

```
$ pki -c Secret123 client-init
------------------
Client initialized
------------------
```

注記

この操作は管理者にとっては任意です。管理者が新しいサブシステムを作成すると、クライアントセキュリティーデータベースが自動的に作成されます。

ローカル証明書の一覧表示

pki client-cert-find

クライアントセキュリティーデータベース内のすべての証明書を一覧表示します。

証明書および秘密鍵のインポート

pki client-cert-import

PKCS #12 ファイルから CA 証明書またはクライアント証明書をインポートします。

例2.1 CA サーバーからの CA 証明書のインポート

CA サーバーから CA 証明書をダウンロードしてインポートするには、以下を実行します。

```
$ pki -c Secret123 -n "CA Signing Certificate - EXAMPLE" client-cert-import --ca-server
---------------------------------------------------------
Imported certificate "CA Signing Certificate - EXAMPLE"
---------------------------------------------------------
```

例2.2 ファイルからの CA 証明書のインポート

ファイルから CA 証明書をインポートするには、以下を実行します。

```
$ pki -c Secret123 -n "CA Signing Certificate - EXAMPLE" client-cert-import --ca-cert ca.pem
---------------------------------------------------------
Imported certificate "CA Signing Certificate - EXAMPLE"
---------------------------------------------------------
```

注記

CA 証明書のインポートは任意です。コマンドラインから SSL 経由でサーバーに接続するときに CA 証明書がクライアントセキュリティーデータベースに存在しない場合は、CA サーバーから CA 証明書をダウンロードしてインポートするかどうかが尋ねられます。

例2.3 クライアント証明書および秘密鍵のインポート

PKCS #12 から秘密鍵をインポートするには、以下を実行します。

```
$ pki -c Secret123 client-cert-import --pkcs12 ca_admin_cert.p12 --pkcs12-password
Secret123
-----------------------------------------
```

```
Imported certificates from PKCS #12 file
----------------------------------------
```

注記

管理者では、証明書と秘密鍵のインポートは任意です。管理者が新しいサブシステムを作成すると、管理者証明書と秘密鍵はクライアントセキュリティーデータベースに自動的に保存されます。

ローカル証明書の削除

pki client-cert-del

ローカル証明書を削除します。

2.4.2. pki証明書を使用した証明書管理

pki cert-* コマンドを使用すると、CA で証明書および証明書要求を管理できます。これらのコマンドの詳細は、pki-cert(1) の man ページを参照してください。

証明書の一覧表示

pki cert-find

すべての証明書を一覧表示します。

例2.4 有効な証明書のみの一覧表示

有効な証明書のみを一覧表示するには、以下を実行します。

```
$ pki cert-find --status VALID
```

例2.5 検索制約が設定されたファイルに基づく証明書の一覧表示

ファイルで定義された検索制約で証明書を一覧表示するには、以下を実行します。

1. 検索制約を定義する XML ファイルを準備します。ファイルは以下の形式に従う必要があります。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CertSearchRequest>

    <serialNumberRangeInUse>true</serialNumberRangeInUse>
    <serialFrom></serialFrom>
    <serialTo></serialTo>

    <subjectInUse>false</subjectInUse>
    <eMail></eMail>
    <commonName></commonName>
    <userID></userID>
    <orgUnit></orgUnit>
```

```
    <org></org>
    <locality></locality>
    <state></state>
    <country></country>

    <matchExactly>false</matchExactly>

    <status></status>

    <revokedByInUse>false</revokedByInUse>
    <revokedBy></revokedBy>

    <revokedOnFrom>false</revokedOnFrom>
    <revokedOnTo></revokedOnTo>

    <revocationReasonInUse>false</revocationReasonInUse>
    <revocationReason></revocationReason>

    <issuedByInUse>false</issuedByInUse>
    <issuedBy></issuedBy>

    <issuedOnInUse>false</issuedOnInUse>
    <issuedOnFrom></issuedOnFrom>
    <issuedOnTo></issuedOnTo>

    <validNotBeforeInUse>false</validNotBeforeInUse>
    <validNotBeforeFrom></validNotBeforeFrom>
    <validNotBeforeTo></validNotBeforeTo>

    <validNotAfterInUse>false</validNotAfterInUse>
    <validNotAfterFrom></validNotAfterFrom>
    <validNotAfterTo></validNotAfterTo>

    <validityLengthInUse>false</validityLengthInUse>
    <validityOperation></validityOperation>
    <validityCount></validityCount>
    <validityUnit></validityUnit>

    <certTypeInUse>false</certTypeInUse>
    <certTypeSubEmailCA></certTypeSubEmailCA>
    <certTypeSubSSLCA></certTypeSubSSLCA>
    <certTypeSecureEmail></certTypeSecureEmail>

</CertSearchRequest>
```

2. コマンドにファイルパスを追加して、**pki cert-find** コマンドを実行します。

```
$ pki cert-find --input filename
```

証明書の表示

pki cert-show

指定した証明書を表示または取得します。

例2.6 証明書のダウンロード

pki cert-show を使用して証明書をダウンロードするには、次のコマンドを実行します。

```
$ pki cert-show certificate ID --encoded --output filename
```

証明書要求の作成

pki cert-request-profile-show および pki cert-request-submit

これらのコマンドは、証明書要求を作成して送信するために使用できます。

例2.7 証明書要求の作成および送信

pki cert-request-profile-show および pki cert-request-submit を使用して証明書 **要求を作成して送信** するには、以下を実行します。

1. CSR を生成します。

```
$ certutil -R -d security database directory -s subject DN -a
```

2. 以下のコマンドを使用して、プロファイルテンプレートを取得します。

```
$ pki cert-request-profile-show profile --output file
```

3. 出力ファイルを編集し、CSR を **cert_request** 属性に挿入します。以下に例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CertEnrollmentRequest>

...

   <Input id="i1">

...

     <Attribute name="cert_request_type">
       <Value>pkcs10</Value>

...

     </Attribute>
     <Attribute name="cert_request">
       <Value>
MIIBZTCBzwIBADAmMRAwDgYDVQQKEwdFWEFNUExFMRIwEAYDVQQDEwlUZX
N0IFVz
ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL7hYQp/g4FblKRd3Cjyfh8e
MFGZLbTDZcY+YBxOk43JeqIDLkGZRHpr/84hK4lgISuyXpvz8owKel2jw6q7bP9Z
0D8AGrrJfEvAuMQrAJiMd/O3U6CKF9+U/z8RjzHPXjzAKl/cIVpqnPuAQOMWQGmx
HkxmLYZww0hKcc9nl5KPAgMBAAGgADANBgkqhkiG9w0BAQUFAAOBgQCtpV2ts1Hp

w+s7ev90d2gRpmPBtNGfOz4OsOpNYbDX3fGabkLFIJAWQ8arjQqToGawIh0nZpND
```

```
UJ9hSa1gIfI+4uxYKjk6cFQAPnZeVgLg1KgELVIzYZ0Qem5NXHmRsR/Vwxh5abzX
XeuHTCnFT0Elpva9mnR+tqe1agZwHghDwQ==
        </Value>

...

      </Attribute>
    </Input>

...

</CertEnrollmentRequest>
```

4. **pki cert-request-submit** コマンドを使用して要求を送信します。

```
$ pki cert-request-submit filename
```

証明書要求ステータスの確認

pki cert-request-show
証明書要求のステータスを表示します。

証明書要求の管理

重要

証明書リクエストの表示または処理は、エージェントの認証情報を使用して実行する必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki cert-request-find
すべての証明書要求を表示します。

pki cert-request-review
証明書要求を確認し、approve または reject などのアクションを実行します。

例2.8 pki cert-requestを使用した証明書の確認

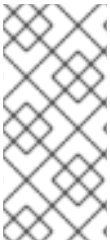
pki cert-request-review を使用して証明書を確認するには、次のコマンドを実行します。

1. 指定の証明書要求で ファイルを生成します。

```
$ pki agent authentication cert-request-review request_ID --output filename
```

2. 生成された出力ファイルを手動で確認し、必要に応じて編集します。
3. コマンドラインに以下のアクションの 1 つを入力してレビューを完了します。
   - approve

- reject
- cancel
- update
- validate
- assign
- unassign



注記

--action を使用して、必要なレビューアクションを直接コマンドに渡すことで、承認プロセスを 1 つのステップで実行することができます。以下に例を示します。

```
$ pki agent authentication cert-request-review request_ID --action approve
```

証明書の取り消し

重要

証明書の取り消し、保持、またはリリースは、エージェントの認証情報を使用して実行する必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki cert-revoke

証明書の取り消し

pki cert-hold

一時的に証明書を保持します。

pki cert-release-hold

保持されている証明書を解放します。

2.4.3. pki ユーザーおよび pki グループを使用したユーザー および グループの管理

pki user-* コマンドおよび **pki group-*** コマンドを使用すると、ユーザーおよびグループを管理できます。これらのコマンドでは、操作が適用されるサブシステムを指定する必要があります。これらのコマンドの詳細は、pki-user(1) および pki-group(1) の man ページを参照してください。

重要

これらのコマンドはすべて、管理者の認証情報を使用して実行する必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki *subsystem*-user-find

ユーザーを一覧表示します。

pki *subsystem*-group-find
グループを一覧表示します。

pki *subsystem*-user-show
指定したユーザーの詳細を表示します。

pki *subsystem*-group-show
指定したグループの詳細を表示します。

pki *subsystem*-user-add
新しいユーザーを追加します。

pki *subsystem*-group-add
新規グループを追加します。

pki *subsystem*-user-mod
既存のユーザーエントリーを変更します。

pki *subsystem*-group-mod
既存のグループエントリーを変更します。

pki *subsystem*-user-del
ユーザーを削除します。

pki *subsystem*-group-del
グループを削除します。

2.4.4. pki group-member および pki user-membershipを使用したグループメンバーおよびユーザーメンバーシップの管理

pki group-member-* コマンド
グループメンバー管理用のコマンド

pki user-membership-* コマンド
ユーザーメンバーシップ管理のコマンド

利用可能なグループメンバーおよびユーザーメンバーシップの管理コマンドの完全なリストについては、**pki group-member** または **pki user-membership** を実行します。コマンドの詳細は、pki-group-member(1) および pki-user-membership(1) の man ページを参照してください。

2.4.5. pki セキュリティードメインを使用したセキュリティードメイン管理

pki securitydomain-show
セキュリティードメイン情報を表示します。このコマンドの詳細は、pki-securitydomain(1) の man ページを参照してください。

2.4.6. pki key-*を使用したキー管理

pki key-* コマンドを使用すると、KRA でキーを管理できます。これらのコマンドの詳細は、 pki-key(1) の man ページを参照してください。

テンプレート

pki key-template-find

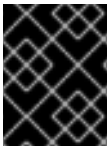
利用可能なすべてのキーテンプレートを一覧表示します。

pki key-template-show

キーテンプレートを表示するか、キーテンプレートをファイルに保存します。

> **例2.9 キーテンプレートのファイルへの保存**
>
> キーテンプレートをファイルに保存するには、以下を実行します。
>
> ```
> $ pki key-template-show retrieveKey --output retrieveKey.xml
> ```

主要なリクエスト



重要

すべてのキー要求は、KRA エージェントの認証情報で実行する必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki key-request-find

送信されたすべてのキー要求を一覧表示します。

pki key-request-show

指定したキーリクエストを表示します

pki key-request-review

キー要求を確認します。レビュープロセスは、例2.8「**pki cert-request**を使用した証明書の確認」で説明されているように、証明書要求の確認と同じルールに従います。

鍵

重要

すべてのキー操作は、KRA エージェントの認証情報で実行する必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki key-find

アーカイブされたすべての鍵を一覧表示します。

pki key-generate

サーバーで新しいキーを生成します。

pki key-archive

コマンドラインで指定されたシークレットをアーカイブする

テンプレートですでに暗号化されたシークレットをアーカイブするには、以下を実行します。

```
$ pki -d ~/.dogtag/pki-tomcat/ca/alias/ -c Secret123 -n caadmin key-archive --input archiveKey.xml
```

pki key-retrieve

キーを取得します。

例2.10 ランダムセキュリティーパラメーターを使用した鍵の取得

無作為に生成されるセキュリティーパラメーターでキーを取得するには、以下を実行します。

```
$ pki -d ~/.dogtag/pki-tomcat/ca/alias/ -c Secret123 -n caadmin key-retrieve --keyID 0x1

Retrieve Key Information
------------------------
  Key Algorithm: RSA
  Key Size: 1024
  Nonce data: rYkeh4Rb+MI=

  Actual archived data:
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBALTyleypbSGRnb8+
P/BItA74mTdLX4eFY+fKE4hraeOV4ts+4M9qfry/FJkbMq3dpIpsxuMmGclbHEUQ
J/MfLAHgaxwVLGK8qCGb0IeY0Z7qIbGucSCLcDVpODlsTvqftK/SJZm56ODu7xXh
...
```

例2.11 カスタムセキュリティーパラメーターによるキーの取得

テンプレートで指定されたカスタムセキュリティーパラメーターでキーを取得するには、以下を実行します。

```
$ pki -d ~/.dogtag/pki-tomcat/ca/alias/ -c Secret123 -n caadmin key-retrieve --input
retrieveKey.xml
```

pki key-recover

キーを復元します。

pki key-show

指定したキーの詳細を表示します。

例2.12 キー ID を指定する際のキーの表示

キー ID を指定するときにキーを表示するには、次のコマンドを実行します。

```
$ pki -d ~/.dogtag/pki-tomcat/ca/alias/ -c Secret123 -n caadmin key-show 0x1
```

```
  Key ID: 0x1
  Client Key ID: test
  Status: active
  Algorithm: RSA
  Size: 1024
  Owner: kraadmin
  Public Key:

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC08pXsqW0hkZ2/Pj/wSLQO+Jk3
S1+HhWPnyhOIa2njleLbPuDPan68vxSZGzKt3aSKbMbjJhnJWxxFECfzHywB4Gsc
FSxivKghm9CHmNGe6iGxrnEgi3A1aTg5bE76n7Sv0iWZuejg7u8V4QmU+jBc79O4
ydfTGLzZvtTVrYbgdQIDAQAB
```

例2.13 クライアントキー ID を指定する際のキーの表示

クライアントキー ID を指定するときにキーを表示するには、次のコマンドを実行します。

```
$ pki -d ~/.dogtag/pki-tomcat/ca/alias/ -c Secret123 -n caadmin key-show --clientKeyID test
  Key ID: 0x1
  Client Key ID: test
  Status: active
  Algorithm: RSA
  Size: 1024
  Owner: kraadmin
  Public Key:

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC08pXsqW0hkZ2/Pj/wSLQO+Jk3
S1+HhWPnyhOIa2njleLbPuDPan68vxSZGzKt3aSKbMbjJhnJWxxFECfzHywB4Gsc
FSxivKghm9CHmNGe6iGxrnEgi3A1aTg5bE76n7Sv0iWZuejg7u8V4QmU+jBc79O4
ydfTGLzZvtTVrYbgdQIDAQAB
```

pki key-mod --status *active*

キーをアクティブにします。**--status** オプションを **inactive** に設定すると、キーが無効になります。

2.4.7. pki ca-kraconnectorを使用した KRA コネクター管理

pki ca-kraconnector-* コマンドを使用すると、KRA コネクターを管理できます。

重要

pki ca-kraconnector-* コマンドはすべて CA に送信され、管理者として実行される必要があります。**pki** コマンドの使用時に認証する方法は、「認証」 を参照してください。

pki ca-kraconnector-show

KRA コネクターを表示します。

pki ca-kraconnector-add

新しい KRA コネクターを追加します。

pki ca-kraconnector-del
KRA コネクターを削除します。

2.4.8. pki caを使用した CA 管理

pki ca-* コマンドを使用すると、さまざまな CA サービスにアクセスできます。

プロファイルの一覧表示

pki ca-profile-find
指定されたデータベース内のすべての CA プロファイルを一覧表示します。

プロファイルの表示

pki ca-profile-show
データベースで指定されたプロファイルを表示します

2.4.9. pki tpsを使用した TPS 管理

pki tps-* コマンドを使用すると、さまざまな TPS サービスにアクセスできます。

アクティビティー

tps-activity-find
すべての TPS アクティビティーを表示します

tps-activity-show
指定したアクティビティーを表示します

Audit

tps-audit-mod
監査設定を変更します。

tps-audit-show
監査設定をファイルに表示します。

Users

pki tps-user-find
すべての TPS ユーザーを表示します。

pki tps-user-show
指定した TPS ユーザーを表示します

pki tps-user-add
新しい TPS ユーザーを追加します。

pki tps-user-mod

既存の TPS ユーザーの変更

pki tps-user-del

TPS ユーザーを削除します。

プロファイル

pki tps-profile-find

すべての TPS プロファイルを表示します。

pki tps-profile-show

指定した TPS ユーザーを表示します

pki tps-profile-add

新しい TPS プロファイルを追加します。

pki tps-profile-mod

既存の TPS プロファイルの変更

pki tps-profile-del

TPS プロファイルを削除します。

第3章 TOKENINFO （外部ハードウェアトークンの管理）

このツールは、Certificate System サブシステムに表示される外部ハードウェアトークンを決定するために使用されます。これは、トークンを使用した問題が、それを検出できない Certificate System に関連するかどうかを診断するために使用できます。

3.1. SYNTAX

TokenInfo ツールの構文は次のとおりです。

```
TokenInfo /directory/alias
```

オプション	説明
/directory/alias	証明書およびキーデータベースディレクトリーへのパスおよびファイルを指定します（例：**/var/lib/pki-ca/alias** ）。

第4章 SSLGET (HTTPS を介したファイルのダウンロード)

このツールは、HTTP 経由でファイルをダウンロードする **wget** コマンドに似ています。**sslget** は、NSS ライブラリーを使用したクライアント認証をサポートします。設定ウィザードは、このユーティリティーを使用して CA からセキュリティードメイン情報を取得します。

4.1. SYNTAX

sslget ツールの構文は次のとおりです。

sslget [-e *プロファイル情報*] -n *rsa_nickname* [[-p *password*] | [-w *passwordFile*]] [-d *dbdir*] [-v] [-V] -r *url hostname* [:*port*]

オプション	説明
e	**オプション:**フォーム名とフォームフィールドを指定して、サブシステムフォームから情報を送信します。たとえば、これを使用して、証明書プロファイルを介して証明書の登録を送信できます。
n	CA 証明書のニックネームを指定します。
p	証明書データベースのパスワードを指定します。**-w** オプションが使用されている場合は使用されません。
w	**オプション:**パスワードファイルのパスと名前を指定します。**-p** オプションが使用されている場合は使用されません。
d	**オプション:**セキュリティーデータベースへのパスを指定します。
v	**オプション:**詳細モードで操作を設定します。
V	**オプション:sslget** ツールのバージョンを指定します。
r **url**	情報のダウンロード元となるサイトまたはサーバーの URL を指定します。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。
hostname	要求を送信するサーバーのホスト名を指定します。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。
port	**オプション:**サーバーのポート番号を指定します。

4.2. 使用方法

sslget を使用して、Certificate System サブシステムに情報を安全に送信できます。たとえば、の証明書プロファイル登録を介して証明書要求を CA に送信するには、コマンドは以下のようになります。

```
sslget -e "profileId=caInternalAuthServerCert&cert_request_type=pkcs10
&requestor_name=TPS-server.example.com-7889
&cert_request=MIIBGTCBxAIBADBfMSgwJgYDVQQKEx8yMDA2MTEwNngxMi
BTZmJheSBSZWRoYXQgRG9tYWluMRIwEAYDVQQLEwlyaHBraS10cHMxHzAdBgNVBA
MTFndhdGVyLnNmYmF5LnJlZGhhdC5jb20wXDANBgkqhkiG9w0BAQEFAANLADBIAk
EAsMcYjKD2cDJOeKjhuAiyaC0YVh8hUzfcrf7ZJlVyROQx1pQrHiHmBQbcCdQxNz
YK7rxWiR62BPDR4dHtQzj8RwIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQAKpuTYGP
%2BI1k50tjn6enPV6j%2B2lFFjrYNwlYWBe4qYhm3WoA0tIuplNLpzP0vw6ttIMZ
kpE8rcfAeMG10doUpp
&xmlOutput=true&sessionID=-4771521138734965265
&auth_hostname=server.example.com&auth_port=9444"
 -d "/var/lib/pki-tps/alias" -p "password123" -v -n "Server-Cert cert-pki-tps" -r "/ca/ee/ca/profileSubmit"
server.example.com:9444
```

第5章 AUDITVERIFY（監査ログ検証）

AuditVerify ツールは、署名された監査ログが秘密鍵で署名され、監査ログが侵害されていないことを確認するために使用されます。

監査担当者は、**AuditVerify** ツールを使用して、署名済み監査ログの信頼性を検証できます。このツールは、署名された監査ログ署名証明書の公開鍵を使用して、署名済み監査ログファイルに埋め込まれたデジタル署名を検証します。ツールの応答は、署名された監査ログが正常に検証されたか、署名済み監査ログが正常に検証されなかったことを示します。検証に失敗すると、署名が検証に失敗したことを監査者に警告します。これは、ログファイルが で改ざんされている可能性があることを示します（侵害）。

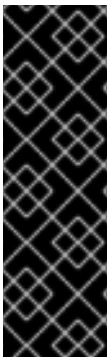
5.1. AUDITOR のデータベースの設定

AuditVerify は、署名された監査ログ署名証明書とそのチェーンを含む一連のセキュリティーデータベース（通常は監査者の個人セキュリティーデータベース）にアクセスする必要があります。発行チェーンの CA 証明書の 1 つは、データベースで信頼できるものとしてマークする必要があります。

監査担当者は、**AuditVerify** を実行する前に、監査署名証明書を個人証明書およびキーデータベースにインポートする必要があります。監査人は、署名済み監査ログファイルを生成した Certificate System インスタンスのセキュリティーデータベースを使用しないでください。簡単にアクセス可能な証明書およびキーデータベースがない場合、監査担当者は証明書およびキーデータベースのセットを作成し、署名された監査ログ署名証明書チェーンをインポートする必要があります。

注記

サブシステムが保持する **signedAudit** ディレクトリーは、監査人を含め、どのユーザーがも書き込み可能ではありません。



重要

auditor ユーザーは以下のいずれかのメンバーである必要があります。

- **pkiaudit** グループ。**/etc/pki/default.cfg** ファイルの **[DEFAULT]** セクションの下にある ***pki_audit_group*** 変数のデフォルト値です。
- **pkispawn** ユーティリティーを実行してサブシステムを作成するときに ***pki_audit_group*** 変数を上書きして、監査グループとして識別されたシステムグループ。

セキュリティーデータベースを作成し、証明書チェーンをインポートするには、以下を実行します。

1. 検証の実行に使用する auditor のホームディレクトリーに特別なディレクトリーを作成します。以下に例を示します。

```
mkdir ~jsmith/auditVerifyDir
```

2. **certutil** ツールを使用して、監査者のホームディレクトリーに空の証明書データベースのセットを作成します。

```
certutil -d ~jsmith/auditVerifyDir -N
```

3. CA の **Retrieval** ページから CA 証明書をダウンロードします。

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

4. CA 証明書をインポートし、署名証明書をデータベースにログインし、CA 証明書を信頼済みとしてマークします。証明書は ASCII 形式で CA から取得できます。

   CA 証明書が **cacert.txt** という名前のファイルにあり、ログ署名証明書が **logsigncert.txt** という名前のファイルにある場合、**certutil** を使用して、これらのファイルを参照する新しい監査セキュリティーデータベースディレクトリーの信頼を設定します。

```
certutil -d ~jsmith/auditVerifyDir/ -A -n "CA Certificate" -t "CT,CT,CT" -a -i
/var/lib/instance_ID/alias/cacert.txt

certutil -d ~jsmith/auditVerifyDir -A -n "Log Signing Certificate"-t ",,P" -a -i
/var/lib/instance_ID/alias/logsigncert.txt
```

5.2. SYNTAX

AuditVerify ツールの構文は以下のとおりです。

```
AuditVerify -d dbdir -n signing_certificate_nickname -a logListFile [-P cert/key_db_prefix] [-v]
```

オプション	説明
a	検証する署名付き監査ログのコンマ区切りリスト（時系列順）を含むテキストファイルを指定します。**logListFile** の内容は、監査ログへの完全パスです。以下に例を示します。 `/var/log/pki-ca/signedAudit/ca_cert-ca_audit, /var/log/pki-ca/signedAudit/ca_cert-ca_audit.20030227102711, /var/log/pki-ca/signedAudit/ca_cert-ca_audit.20030226094015` このファイルは、**~jsmith/auditDir** などの特別な監査ディレクトリーなど、監査人が書き込み可能なディレクトリーに作成する必要があります。
d	インポートされた監査ログ署名証明書を使用して、セキュリティーデータベースを含むディレクトリーを指定します。このディレクトリーはほぼ常に、**~jsmith/auditVerifyDir/** などの個人ディレクトリー内の監査人自身の個人証明書データベースです。
n	ログファイルの署名に使用される証明書のニックネームを指定します。ニックネームは、ログ署名証明書がそのデータベースにインポートされたときに使用されたものです。

オプション	説明
P	**オプション:**証明書およびキーデータベースのファイル名の前に付ける接頭辞。これを使用する場合は、この引数に空の引用符「」（）の値を指定する必要があります。これは、監査人は Certificate System インスタンスとは別の証明書およびキーデータベースを使用するため、接頭辞を新しい監査セキュリティーデータベースファイルの前に付ける必要があるためです。
v	**オプション:**詳細出力を指定します。

5.3. 戻り値

AuditVerify を使用すると、以下のいずれかのコードが返されます。

戻り値	説明
0	署名された監査ログが正常に検証されたことを示します。
1	ツールの実行中にエラーが発生したことを示します。
2	指定されたファイルに 1 つ以上の無効な署名が見つかったことを示します。つまり、少なくとも 1 つのログファイルを検証できませんでした。

5.4. 使用方法

別の監査データベースディレクトリーを設定したら、以下を実行します。

1. 検証するログファイルのコンマ区切りリストを含むテキストファイルを作成します。このファイルの名前は、**AuditVerify** コマンドで参照されます。

   たとえば、このファイルは **/etc/audit** ディレクトリーの **logListFile** になります。この内容は、"**auditlog. 1213**,**auditlog.1215**." など、検証される監査ログのコンマ区切りリストです。

2. 監査データベースに接頭辞が含まれておらず、**/home/smith/.mozilla** などのユーザーのホームディレクトリーにあり、署名証明書のニックネームが **auditsigningcert** の場合、**AuditVerify** コマンドは次のように実行されます。

   ```
   AuditVerify -d ~jsmith/auitVerifyDir -n auditsigningcert -a /etc/audit/logListFile -P "" -v
   ```

5.5. 結果

入力ファイル **audit_list** は、検証される監査ログへのフルパスを提供する簡単なテキストファイルです。

–

```
cat ~jsmith/auditVerifyDir/audit_list
 /var/lib/pki-ca/logs/signedAudit/ca_audit.20110211145833
```

いずれかのファイルに変更が加えられていない場合、**AuditVerify** はすべての署名が有効であるというメッセージを返します。

```
AuditVerify -d ~jsmith/auditVerifyDir -n "Log Signing Certificate" -a  ~jsmith/auditVerifyDir/audit_list

Verification process complete.
Valid signatures: 20
Invalid signatures: 0
```

ログファイルへの変更がある場合、署名は無効になります。この場合、**AuditVerify** は無効な署名があり、編集したログファイルの名前と変更の行番号を返すことを示しています。

```
AuditVerify -d ~jsmith/auditVerifyDir -n "Log Signing Certificate" -a  ~jsmith/auditVerifyDir/audit_list
======
File:
/var/lib/pki-ca/logs/signedAudit/ca_audit.20110211145833
======
Line 52: VERIFICATION FAILED: signature of /var/lib/pki-
ca/logs/signedAudit/ca_audit.20101213141439:48 to /var/lib/pki-
ca/logs/signedAudit/ca_audit.20101213141439:51

Verification process complete.
Valid signatures: 19
Invalid signatures: 1
```

第6章 SETPIN （エンティティーの一意の PIN の生成）

Certificate System が **UidPwdPinDirAuth** 認証プラグインモジュールを使用するには、認証ディレクトリーに、証明書を発行する各エンドエンティティーの一意の PIN が含まれている必要があります。Certificate System は、LDAP ディレクトリーにエンドエンティティーエントリーの一意の PIN を生成するツール(**PIN Generator**)を提供します。このツールは、これらの PIN をハッシュ化された値として、対応するユーザーエントリーに対して同じディレクトリーに保存します。また、PIN をテキストファイルにコピーし、PIN をエンドエンティティーに送信できるようにします。

6.1. SETPIN コマンド

本章では、**setpin** ツールの構文および引数と予想される応答について説明します。

6.1.1. setpin.conf 設定ファイルの編集

setpin ツールは、設定ファイル **setpin.conf** を使用して、必要なオプションの一部を保存できます。**setpin** を実行する前に、このファイルを変更してディレクトリー情報を反映し、以下のコマンドを実行して **setpin** ツールがこのファイルを使用するように設定します。

1. **setpin.conf** ファイルを開きます。

```
cd /usr/lib/pki/native-tools
vi setpin.conf
```

2. ディレクトリーのインストール情報に一致するように、ファイルのディレクトリーパラメーターを編集します。

```
#------- Enter the hostname of the LDAP server
host=localhost

#------- Enter the port number of the LDAP server
port=389

#------- Enter the DN of the Directory Manager user
binddn=CN=Directory Manager

#------- Enter the password for the Directory manager user
bindpw=

#    Enter the DN and password for the new pin manager user
pinmanager=cn=pinmanager,dc=example,dc=com
pinmanagerpwd=

#    Enter the base over which this user has the power
#    to remove pins
basedn=ou=people,dc=example,dc=com

## This line switches setpin into setup mode.
## Please do not change it.
setup=yes
```

3. **setpin** を実行し、オプションファイルを **setpin.conf** に設定します。

```
setpin optfile=/usr/lib/pki/native-tools/setpin.conf
```

6.1.2. Syntax

setpin の構文は以下のとおりです。

setpin host=*host_name* [port=*port_number*] binddn=*user_id* [bindpw=*bind_password*] filter="*LDAP_search_filter*" [basedn=*LDAP_base_DN*] [[length=*PIN_length*] | [minlength=*minimum_PIN_length*] | [maxlength=*maximum_PIN_length*]] [gen=*character_type*] [case=upperonly] [hash=*algorithm*] [saltattribute=*LDAP_attribute_to_use_for_salt_creation*] [input=*file_name*] [output=*file_name*] [write] [clobber] [testpingen=*count*] [debug] [optfile=*file_name*] [setup [pinmanager=*pinmanager_user*] [pinmanagerpwd=*pinmanager_password*]]

オプション	説明
ホスト	**必須。**接続する LDAP ディレクトリーを指定します。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。
port	バインドする LDAP ディレクトリーポートを指定します。デフォルトのポート番号はデフォルトの LDAP ポート **389** です。
binddn	**必須。**PIN Generator が LDAP ディレクトリーにバインドするユーザーを指定します。このユーザーアカウントには、ディレクトリーへの読み取り/書き込みアクセスが必要です。
bindpw	**binddn** オプションに設定されたユーザー ID のパスワードを指定します。コマンドラインでバインドパスワードが指定されていない場合、ツールはこれを要求します。
filter	**必須。**ツールが PIN を生成するディレクトリー内の DN の検索フィルターを設定します。
basedn	DN を検索するベース DN を指定します。この引数を指定しないと、フィルターはルートから検索します。
長さ	PIN に含める必要のある正確な番号を指定します。デフォルトは 6 です。**minlength** または **maxlength** では を使用しないでください。
minlength	生成される PIN の最小長を設定します。**maxlength** と併用すると、PIN 長さの範囲の下限が設定されます。**長さ** で を使用しないでください。

オプション	説明
maxlength	生成される PIN の最大長を設定します。**minlength** と併用すると、PIN 長さの範囲の上限が設定されます。**長さ** で を使用しないでください。
Gen	PIN の文字タイプを指定します。パスワードの 文字は、アルファベット文字(RNG-alpha)、英数字(**RNG-alphanum**)、または任意の印刷可能な ASCII 文字(**printableascii**)から作成できます。
ケース	キャラクターケースを大文字のみに制限します。それ以外の場合は、大文字と小文字が混在しています。アルファベット文字を大文字に制限すると、パスワード領域の全体的な組み合わせが大幅に削減されます。**gen** の **ユースケース**。
ハッシュ	認証ディレクトリーに保存する前に PIN をハッシュ化するメッセージダイジェストアルゴリズムを指定します。 **注記** Directory Server は、受信ハッシュ化されたパスワードに制限がある可能性があるため、これを **none**（ハッシュ化されない）に設定する必要があります。 デフォルトは **sha1** で、160 ビットのメッセージダイジェストを生成します。**md5** は 128 ビットのメッセージダイジェストを生成します。**none** は PIN をハッシュしません。
saltattribute	ソルトの作成に使用する LDAP 属性を指定します。これは ***dn*** に設定する必要があります。属性が設定されている場合、ツールは属性の値を各 PIN に統合し、生成される文字列をハッシュルーチンでハッシュします。詳細は、「PIN がディレクトリーに保存される方法」 を参照してください。 ***hash*** の値が **none** に設定されている場合、この属性は無視されます。これが推奨される設定です。
入力 (input)	処理する DN の一覧を含むファイルを指定します。これを使用すると、ツールはフィルターされた DN を入力ファイル内の DN と比較して、それらの DN のみに PIN を生成します。
出力 (output)	**setpin** が PIN を生成するため、PIN を書き込むファイルへの絶対パスを指定します。ファイルが設定されていない場合、出力は標準出力に書き込まれます。出力ファイルが設定されているかどうかにかかわらず、すべてのエラーメッセージが標準エラーに転送されます。

オプション	説明
write	ツールが PIN をディレクトリーに書き込むかどうかを設定します。指定した場合、PIN は生成されるとディレクトリーに書き込まれます。それ以外の場合は、ツールはディレクトリーに変更を加えません。PIN をチェックする場合は、ディレクトリーに PIN を書き込まないでください。PIN は出力ファイルで表示して、正しいユーザーに割り当てられていること、および長さと文字の制限に準拠していることを確認できます。詳細は、「Output File」 を参照してください。
clobber	DN に関連付けられている既存の PIN（存在する場合）を上書きします。このオプションを使用しない場合、既存の PIN はディレクトリーに残ります。
testpingen	PIN 生成モードをテストします。**count** は、テスト用に生成する PIN の総数を設定します。
debug	標準エラーにデバッグ情報を書き込みます。**debug=attrs** を指定すると、ツールはディレクトリー内の各エントリーに関する詳細情報を書き込みます。
optfile	ファイルからオプションを読み取るためのツールを設定します（行ごとに 1 つずつ）。これにより、コマンドラインで引数を入力せずに、すべての引数をファイルに配置できます。1 つの設定ファイル **setpin.conf** は、**/usr/lib/pki/native-tools** ディレクトリーにあります。
setup	設定モードに切り替えます。これにより、ツールがディレクトリースキーマに追加できるようになります。
pinmanager	**basedn** で指定した PIN を削除するパーミッションを持つ PIN マネージャーユーザーを指定します。**設定** オプションで使用されます。
pinmanagerpwd	PIN マネージャーユーザーのパスワードを指定します。**設定** オプションで使用されます。

6.1.3. 使用方法

まず、 **setpin.conf** ファイルを参照する **optfile** オプションを指定して setpin コマンドを実行します。

```
setpin optfile=/usr/lib/pki/native-tools/setpin.conf
```

このツールは、新しい属性 (デフォルトでは **pin**) および新しいオブジェクトクラス (デフォルトは **pinPerson**) でスキーマを変更し、**pinmanager** ユーザーを作成し、ACI を設定して、**pinmanager** ユーザーのみが **pin** 属性を編集できるようにします。

次に、**setpin** コマンドのセットアップモードを無効にします。**setup** 行をコメントアウトするか、値を no に変更します。

```
vim /usr/lib/pki/native-tools/setpin.conf

setup=no
```

設定が完了したら、**setpin** を使用して PIN を生成できます。

以下のコマンドは、**csldap** という名前の LDAP ディレクトリーの識別名に **CN** 属性を持つすべてのエントリーの PIN を生成します。ポート **389** でリッスンします。PIN Generator は、**Directory Manager** としてディレクトリーにバインドされ、ディレクトリーツリーのベース DN **dn=dc=example,dc=com** からディレクトリーの検索を開始します。既存の PIN は新しい PIN で上書きされます。

```
setpin host=csldap port=389 binddn="CN=directory manager" bindpw=password filter="(cn=*)"
basedn="dc=example,dc=com" clobber write hash=none
```

6.2. SETPIN の仕組み

PIN Generator は、LDAP ディレクトリーにユーザーエントリーの PIN を生成し、これらの PIN でディレクトリーを更新します。**setpin** コマンドを実行するには、次の 5 つのオプションが必要です。

- LDAP サーバーの**ホスト名（ホスト**）および**ポート番号（ポート**）
- バインド DN (**binddn**)およびパスワード(**bindpw**)
- PIN を必要とするユーザーエントリーを**フィルターリングするための LDAP フィルター（フィルター**）

setpin コマンドは以下のようになります。

```
setpin host=csldap port=19000 binddn="CN=Directory Manager" bindpw=secret filter="
(ou=employees)" basedn="dc=example,dc=com"
```

この例では、**従業員** 組織単位(**ou**)のすべてのエントリーをクエリーします。フィルターに一致する各エントリーについて、情報は標準エラーおよび標準出力に出力されます。

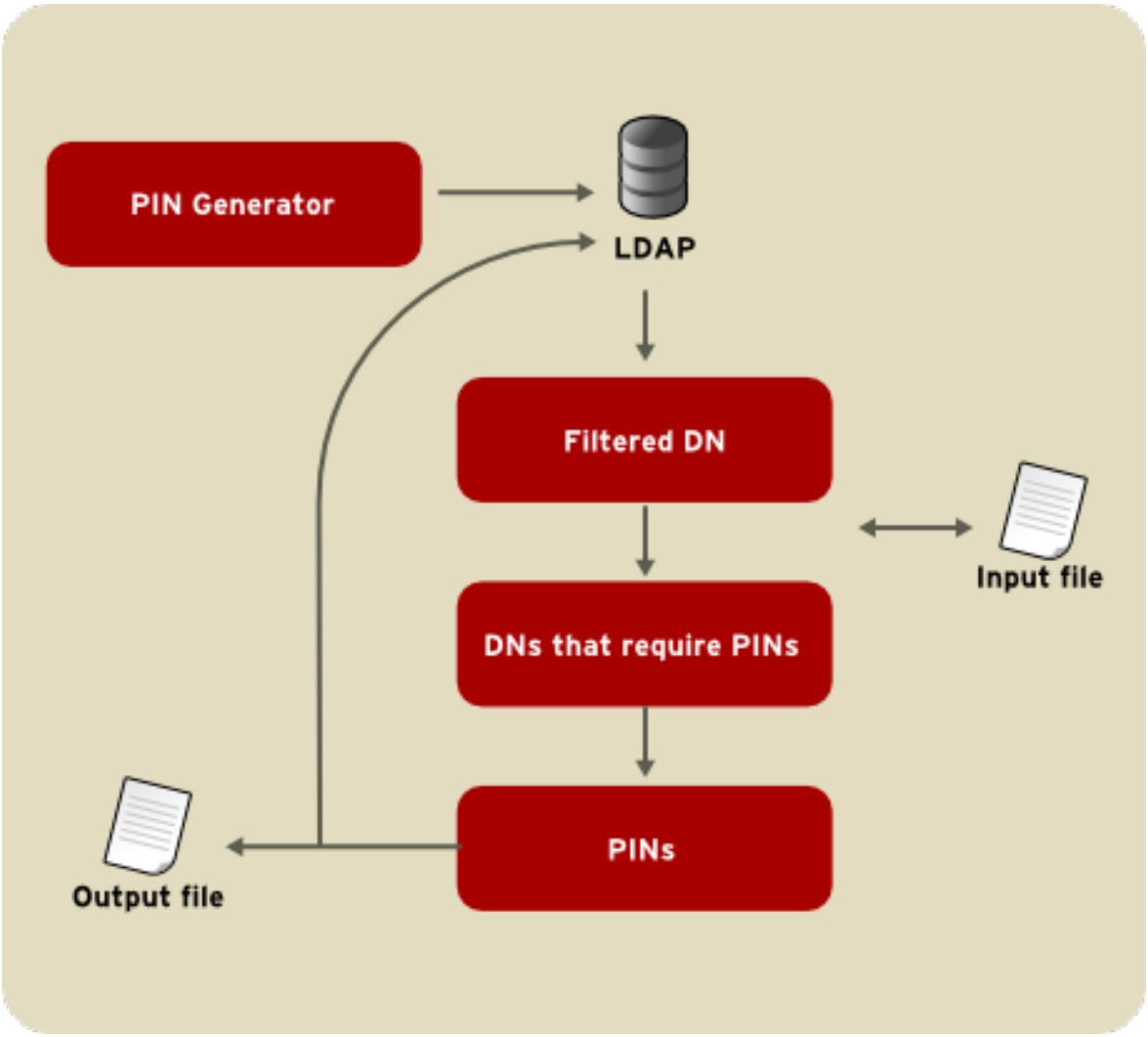
注記

PIN Generator はディレクトリーに多くの変更を行うため、正しいフィルターを使用するか、間違ったエントリーを変更することが重要です。オプションが使用されない限り、ディレクトリーに変更が加えられないため、**write** オプションを使用することは安全です。これにより、エントリーの変更前に PIN を検証できます。

この情報は、出力オプションを使用して別の出力ファイルに書き込むことができます。詳細は、「Output File」 を参照してください。**LDAP 検索フィルターによって返されるエントリーは、エントリー DN を一覧表示する ASCII 入力ファイルを使用してさらに制限できます。ファイル内のエントリーのみが更新されます。入力ファイルは、入力 オプションで設定されます。入力ファイルは LDAP**

ディレクトリーエントリーの代わりではありません。フィルター属性を指定する必要があります。入力ファイルの詳細は、「入力ファイル」 を参照してください。図6.1「PIN の生成時の入出力ファイルの使用」 は、入力ファイルと出力ファイルが setpin ツールでどのように機能するかを示します。

図6.1 PIN の生成時の入出力ファイルの使用



出力ファイルには、次の例に示すように、実行中の setpin からのエントリーと PIN 情報が含まれます。

```
Processing: cn=QA Managers,ou=employees,dc=example,dc=com
Adding new pin/password
dn:cn=QA Managers,ou=employees,dc=example,dc=com
pin:lDWynV
status:notwritten

Processing: cn=PD Managers,ou=employees,dc=example,dc=com
Adding new pin/password
dn:cn=PD Managers,ou=employees,dc=example,dc=com
pin:G69uV7
status:notwritten
```

出力には、ディレクトリー内の各エントリーのステータスも含まれます。ステータスの値は 表6.1「PIN ジェネレーターステータス」 に一覧表示されます。

表6.1 PIN ジェネレーターステータス

終了コード	説明
notwritten	書き込みオプションが使用されていなかったため、PIN はディレクトリーに書き込みされませんでした。
writefailed	ツールはディレクトリーの変更を試みましたが、書き込み操作は失敗しました。
追加済み	このツールは、新しい PIN をディレクトリーに正常に追加されました。
replaced	ツールは古い PIN を新しい PIN に置き換わりました。これは、clobber オプションが使用されていることを意味します。
notreplaced	このツールは、古い PIN を新しい PIN に置き換えませんでした。これは、clobber オプションが使用されていなかったことを意味します。

ユーザーの PIN がすでに存在する場合は、setpin コマンドが 2 回実行される場合は変更されません。これにより、PIN を受け取ったユーザーの PIN を上書きせずに、新規ユーザー用に新しい PIN を作成できます。PIN を上書きするには、clobber オプションを使用します。

フィルターが正しいユーザーと一致していることを確認したら、write オプションを指定して setpin コマンドを再度実行し、出力 はファイルの名前に設定して、所有されていない PIN をキャプチャーします。出力ファイルの詳細は、「Output File」 を参照してください。

6.2.1. 入力ファイル

PIN ジェネレーターは、入力 引数で指定されたテキストファイルで変更する DN の一覧を受け取ることができます。入力ファイルが指定されている場合、ツールはフィルターによって返された DN を入力ファイル内の DN と比較し、入力ファイル内の一致する DN のみを更新します。

入力により、ユーザーは、変更する DN の正確なリストで PIN ジェネレーターを指定できます。また、すべての DN または特定の DN のプレーンテキストの PIN を使用して PIN ジェネレーターを指定することもできます。

入力ファイルを使用する場合は、一般的な状況が 2 つあります。

- ユーザーディレクトリーのすべてのエントリーに PIN が設定され、新しいユーザーが組織に参加している場合。新規ユーザーが証明書を取得するには、ディレクトリーに PIN が含まれている必要があります。他のユーザーエントリーを変更せずに、これら 2 つのエントリーに対してのみピンを生成する必要があります。複雑な LDAP フィルターを作成する代わりに、入力ファイルを使用すると一般的なフィルターを使用でき、変更されたエントリーは入力ファイルにリストされている 2 人のユーザーの DN に制限されます。

- Social Security 番号などの特定の値を PIN として使用する場合は、Social Security 番号を入力ファイルに配置して、それらの番号を PIN ジェネレーターに PIN として指定することができます。これらは、ハッシュ化された値として ディレクトリーに保存されます。

入力ファイルの形式は、status 行を除き、出力ファイル(「Output File」を参照)の形式と同じです。入力ファイルでは、PIN はファイル内のすべての DN、特定の DN の場合は、または DN のいずれも設定できません。DN に PIN 属性がない場合、ツールはランダムな PIN を自動的に生成します。

入力ファイルの例を以下に示します。

```
dn:cn=user1, dc=example,dc=com

dn:cn=user2, dc=example,dc=com

...
dn:cn=user3, dc=example,dc=com
```

プレーンテキスト形式で DN にピンを指定することもできます。これらの PIN はコマンドライン引数に従ってハッシュ化されます。

```
dn:cn=user1, dc=example,dc=com
pin:pl229Ab

dn:cn=user2, dc=example,dc=com
pin:9j65dSf

...
dn:cn=user3, dc=example,dc=com
pin:3knAg60
```

注記

ハッシュ化された PIN はツールに提供できません。

6.2.2. Output File

PIN Generator は、出力オプションで指定されたテキストファイルに 出力 をキャプチャできます。

出力には、以下の形式のレコードシーケンスが含まれます。

```
dn: user_dn1
pin: generated_pin1
status: status1

dn: user_dn2
pin: generated_pin2
status: status2

...
dn: user_dn#
pin: generated_pin#
status: status#
```

user_dn は、DN フィルターに一致する、または入力ファイルに一覧表示されている識別名です。デフォルトでは、区切り文字はセミコロン(;)またはコマンドラインで定義された文字です。generated_pin は、使用される長さパラメーターに応じて固定または変数の長さの文字列です。status は、表6.1「PIN ジェネレーターステータス 」 に一覧表示されている値の 1 つです。

各レコードの最初の行は常に DN です。ピン と ステータス の後続の行は任意です。このレコードは、Unix の行末シーケンス(\n)を使用して空の行で終わります。

6.2.3. PIN がディレクトリーに保存される方法

各 PIN は **saltattribute** 引数で指定された対応する LDAP 属性と連結されます（デフォルトではエントリー DN、推奨される値です）。この引数を指定しないと、DN が使用されます。

この文字列は、ハッシュ引数で指定されたルーチンでハッシュ化されます。デフォルトのアルゴリズムは SHA-1 ですが、受信ハッシュ化されたパスワードの使用を制限する Directory Server で動作するように、これを none に設定する必要があります。

1つのバイトが追加され、使用されるハッシュタイプが示されます。PIN は以下のように保存されます。

```
byte[0] = X
```

X の値は、PIN の生成プロセス中に選択したハッシュアルゴリズムによって異なります。

X	ハッシュアルゴリズム
0	SHA-1
1	MD5
45	none

PIN は、base-64 でエンコードされた値としてではなく、バイナリー値としてディレクトリーに保存されます。

6.2.4. 終了コード

PIN Generator の実行が終了すると、終了方法を示す結果コードが返されます。これらの結果コードは 表6.2「PIN ジェネレーターによって返される結果コード」 に一覧表示されます。

表6.2 PIN ジェネレーターによって返される結果コード

結果コード	説明
0	PIN の生成は成功しました。指定されたディレクトリー内のすべての DN に PIN が設定されました。
4	このツールは、**binddn** パラメーターで指定されたユーザーとしてディレクトリーにバインドできませんでした。
5	このツールは、**output** パラメーターで指定された出力ファイルを開けませんでした。

結果コード	説明
7	コマンドライン引数の解析中にエラーが発生しました。
8	このツールは、入力パラメーターで指定された入力ファイルを開けませんでした。
9	ツールで内部エラーが発生しました。
10	このツールは、入力ファイルで重複するエントリーを見つけました。
11	このツールは、ディレクトリー内の saltattribute パラメーターで指定された salt 属性 を見つけませんでした。

第7章 ATOB (ASCII からバイナリーへの変換)

Certificate System ASCII からバイナリーツールでは、ASCII base-64 でエンコードされたデータをバイナリーの base-64 でエンコードされたデータに変換します。

7.1. SYNTAX

ASCII からバイナリーツール **AtoB** の構文は次のとおりです。

AtoB *input_file output_file*

オプション	説明
input_file	base-64 でエンコードされた ASCII データへのパスおよびファイルを指定します。
output_file	ユーティリティーがバイナリー出力を書き込むファイルを指定します。

7.2. 使用方法

このサンプルコマンドは、**ascii_data.in** ファイルの base-64 ASCII データを取得し、同等のデータを **binary_data.out** ファイルに書き込みます。

AtoB /usr/home/smith/test/ascii_data.in /usr/home/smith/test/binary_data.out

第8章 BTOA (CONVERTING BINARY TO ASCII)

Certificate System バイナリーを ASCII ツールに、**BtoA** はバイナリー base-64 でエンコードされたデータを ASCII base-64 でエンコードされたデータに変換します。

8.1. SYNTAX

BtoA ツールは、以下の構文を使用します。

```
BtoA input_file output_file
```

オプション	説明
input_file	base-64 でエンコードされたバイナリーデータのパスおよびファイルを指定します。
output_file	ツールが ASCII 出力を書き込むパスとファイルを指定します。

8.2. 使用方法

以下の **BtoA** ユーティリティーの例では、**binary_data.in** ファイルで base-64 でエンコードされたバイナリーデータを取得し、データと同等の ASCII を **ascii_data.out** ファイルに書き込みます。

```
BtoA /usr/home/smith/test/binary_data.in /usr/home/smith/test/ascii_data.out
```

第9章 PRETTYPRINTCERT （印刷証明書）

Pretty Print Certificate ユーティリティー **PrettyPrintCert** は、ASCII base-64 でエンコードされたデータとして保存された証明書の内容を読み取り可能な形式に出力します。

9.1. SYNTAX

PrettyPrintCert コマンドの構文は次のとおりです。

> **PrettyPrintCert [-simpleinfo]** *input_file* **[***output_file***]**

オプション	説明
simpleinfo	オプション:解析が簡単な形式で、限られた証明書情報を表示します。
input_file	ASCII base-64 でエンコードされた証明書を含むファイルへのパスを指定します。
output_file	オプション:ツールが証明書を書き込むパスとファイルを指定します。このオプションを指定しないと、証明書情報は標準出力に書き込まれます。

9.2. 使用方法

以下の例では、ASCII base-64 でエンコードされた証明書を **ascii_cert.in** ファイルで変換し、証明書を pretty-print 形式で出力ファイル **ascii_cert.out** に書き込みます。

```
PrettyPrintCert /usr/home/smith/test/ascii_cert.in /usr/home/smith/test/ascii_cert.out
```

ascii_cert.in の base-64 でエンコードされた証明書データは以下のようになります。

```
-----BEGIN CERTIFICATE-----
MIIC2DCCAkGgAwIBAgICEAwwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMxIzA
hBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0wGwYDVQQLExRXaWRnZX
QgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVzdCBUZXN0I
FRlc3QgQ0EwHhcNOTkwMjE4MDMMzM5WhcNMDAwMjE4MDM0MzM5WjCBrjELMAkGA1UEB
hMCVVMxJjAkBgNVBAoTHU5ldHNjYXBlIENvbW11bmljYXRpb25zIENvcnAuMRUwEwYD
VQQLEwOZXRzY2FwZSBDTVMxGDAWBEBEwhtaGFybXNlbjEfMB0GA1UEAxWaW50ZGV2Y2
EgQWRtaW5pcwp0frfJOObeiSsia3BuifRHBNw95ZZQR9NIXr1x5bE
-----END CERTIFICATE-----
```

ascii_cert.out ファイルの pretty-print 形式の証明書は以下のようになります。

```
Certificate:
Data:
Version: v3
Serial Number: 0x100C
Signature Algorithm: OID.1.2.840.113549.1.1.5 -1.2.840.113549.1.1.5
```

```
Issuer: CN=Test CA,OU=Widget Makers 'R'Us,O=Example Corporation, Widgets\,Inc.,C=US
Validity:
 Not Before: Wednesday, February 17, 1999 7:43:39 PM
 Not After: Thursday, February 17, 2000 7:43:39 PM
Subject: MAIL=admin@example.com,CN=testCA Administrator, UID=admin, OU=IS,
 O=Example Corporation,C=US
Subject Public Key Info:
 Algorithm: RSA - 1.2.840.113549.1.1.1
 Public Key:
   30:81:89:02:81:81:00:DE:26:B3:C2:9D:3F:7F:FA:DF:
   24:E3:9B:7A:24:AC:89:AD:C1:BA:27:D1:1C:13:70:F7:
   96:59:41:1F:4D:21:7A:F5:C7:96:C4:75:83:35:9F:49:
   E4:B0:A7:5F:95:C4:09:EA:67:00:EF:BD:7C:39:92:11:
   31:F2:CA:C9:16:87:B9:AD:B8:39:69:18:CE:29:81:5F:
   F3:4D:97:B9:DF:B7:60:B3:00:03:16:8E:C1:F8:17:6E:
   7A:D2:00:0F:7D:9B:A2:69:35:18:70:1C:7C:AE:12:2F:
   0B:0F:EC:69:CD:57:6F:85:F3:3E:9D:43:64:EF:0D:5F:
   EF:40:FF:A6:68:FD:DD:02:03:01:00:01:
Extensions:
 Identifier: 2.16.840.1.113730.1.1
 Critical: no
 Value: 03:02:00:A0:
Identifier: Authority Key Identifier - 2.5.29.35
 Critical: no
 Key Identifier:
   EB:B5:11:8F:00:9A:1A:A6:6E:52:94:A9:74:BC:65:CF:
 07:89:2A:23:
Signature:
 Algorithm: OID.1.2.840.113549.1.1.5 - 1.2.840.113549.1.1.5
 Signature:
   3E:8A:A9:9B:D1:71:EE:37:0D:1F:A0:C1:00:17:53:26:
   6F:EE:28:15:20:74:F6:C5:4F:B4:E7:95:3C:A2:6A:74:
   92:3C:07:A8:39:12:1B:7E:C4:C7:AE:79:C8:D8:FF:1F:
   D5:48:D8:2E:DD:87:88:69:D5:3A:06:CA:CA:9C:9A:55:
   DA:A9:E8:BF:36:BC:68:6D:1F:2B:1C:26:62:7C:75:27:
   E2:8D:24:4A:14:9C:92:C6:F0:7A:05:A1:52:D7:CC:7D:
   E0:9D:6C:D8:97:3A:9C:12:8C:25:48:7F:51:59:BE:3C:
   2B:30:BF:EB:0A:45:7D:A6:49:FB:E7:BE:04:05:D6:8F:
```

以下のコマンド例は、**ascii_cert.in** ファイルの ASCII base-64 でエンコードされた証明書を取得し、証明書に含まれる情報を単純な形式出力ファイル **cert.simple** に書き込みます。

```
PrettyPrintCert -simpleinfo /usr/home/smith/test/ascii_cert.in /usr/home/smith/test/cert.simple
```

ascii_cert.in ファイルの base-64 でエンコードされた証明書データは以下のようになります。

```
-----BEGIN CERTIFICATE-----
MIIC2DCCAkGgAwIBAgICEAwwDQYJKoZIhvcNAQEFBQAwfDELMAkGA1UEBhMCVVMxIzA
hBgNVBAoTGlBhbG9va2FWaWxsZSBXaWRnZXRzLCBJbmMuMR0wGwYDVQQLExRXaWRnZX
QgTWFrZXJzICdSJyBVczEpMCcGA1UEAxMgVGVzdCBUZXN0IFRlc3QgVGVzdCBUZXN0I
FRlc3QgQ0EwHhcNOTkwMjE4MDMMzM5WhcNMDAwMjE4MDM0MzM5WjCBrjELMAkGA1UEB
hMCVVMxJjAkBgNVBAoTHU5ldHNjYXBlIENvbW11bmljYXRpb25zIENvcnAuMRUwEwYD
```

```
VQQLEwOZXRzY2FwZSBDTVMxGDAWBEBEwhtaGFybXNlbjEfMB0GA1UEAxWaW50ZGV2Y2
EgQWRtaW5pcwp0frfJOObeiSsia3BuifRHBNw95ZZQR9NlXr1x5bE
-----END CERTIFICATE-----
```

cert.simple 出力ファイルの単純な証明書情報は以下のようになります。

```
MAIL=admin@example.com
CN=testCA Administrator
UID=admin
OU=IS
O=Example Corporation
C=US
```

第10章 PRETTYPRINTCRL （読み取り可能な CRL の印刷）

Pretty Print CRL ツール **PrettyPrintCrl** は、ASCII base-64 でエンコードされたファイルに、証明書失効リスト(CRL)の内容を読み取り可能な形式で出力します。

10.1. SYNTAX

PrettyPrintCrl ユーティリティーの構文は以下のようになります。

```
PrettyPrintCrl input_file [output-file]
```

オプション	説明
input_file	ASCII base-64 でエンコードされた CRL を含むファイルへのパスを指定します。
output_file	オプション:CRL を書き込むファイルへのパスを指定します。出力ファイルが指定されていない場合、CRL 情報は標準出力に書き込まれます。

10.2. 使用方法

以下の **PrettyPrintCrl** コマンドの例では、ASCII base-64 でエンコードされた CRL を**ascii_crl.in** ファイルで取得し、CRL を pretty-print 形式で出力ファイル **ascii_crl.out** に書き込みます。

```
PrettyPrintCrl /usr/home/smith/test/ascii_crl.in /usr/home/smith/test/ascii_crl.out
```

ascii_crl.in ファイルの base-64 でエンコードされた CRL は以下のようになります。

```
-----BEGIN CRL-----
MIIBkjCBAIBATANBgkqhkiG9w0BAQQFADAsMREwDwYDVQQKEwhOZXRzY2FwZTEXMBUG
A1UEAxMOQ2VydDQwIFRlc3QgQ0EXDTk4MTIxNzIyMzcyNFowgaowIAIBExcNOTgxMjE
1MTMxODMyWjAMMAoGA1UdFQQDCgEBMCACARIXDTk4MTINTEzMjA0MlowDDAKBgNVHR
U
EAwoBAjAgAgERFw05ODEyMTYxMjUxNTRaMAwwCgYDVR0VBAMKAQEwIAIBEBcNOTgxMj
E3MTAzNzI0WjAMMAoGA1UdFQQDCgEDMCACAQoXDTk4MTEyNTEzMTExOFowDDAKBgNV
H
RUEAwoBATANBgkqhkiG9w0BQQFAAOBgQBCN85O0GPTnHfImYPROvoorx7HyFz2ZsuKs
VbITcemsX0NL7DtOa+MyY0pPrkXgm157JrkxEJ7GBOeogbAS6iFbmeSqPHj8+
-----END CRL-----
```

ascii_crl.out 出力ファイルの pretty-print 形式の CRL は以下のようになります。

```
Certificate Revocation List:
Data:
Version: v2
Signature Algorithm: MD5withRSA - 1.2.840.113549.1.1.4
Issuer: CN=Test CA,O=Example Corporation
This Update: Thu Dec 17 14:37:24 PST 1998
Revoked Certificates:
Serial Number: 0x13
```

```
 Revocation Date: Tuesday, December 15, 1998 5:18:32 AM
 Extensions:
  Identifier: Revocation Reason - 2.5.29.21
  Critical: no
  Reason: Key_Compromise
Serial Number: 0x12
 Revocation Date: Tuesday, December 15, 1998 5:20:42 AM
 Extensions:
  Identifier: Revocation Reason - 2.5.29.21
  Critical: no
  Reason: CA_Compromise
Serial Number: 0x11
 Revocation Date: Wednesday, December 16, 1998 4:51:54 AM
 Extensions:
  Identifier: Revocation Reason - 2.5.29.21
  Critical: no
  Reason: Key_Compromise
Serial Number: 0x10
 Revocation Date: Thursday, December 17, 1998 2:37:24 AM
 Extensions:
  Identifier: Revocation Reason - 2.5.29.21
  Critical: no
  Reason: Affiliation_Changed
Serial Number: 0xA
 Revocation Date: Wednesday, November 25, 1998 5:11:18 AM
 Extensions:
  Identifier: Revocation Reason - 2.5.29.21
  Critical: no
  Reason: Key_Compromise
Signature:
 Algorithm: MD5withRSA - 1.2.840.113549.1.1.4
 Signature:
  42:37:CE:4E:D0:63:D3:9C:77:C8:99:83:D1:3A:FA:28:
  AF:1E:C7:C8:5C:F6:66:CB:8A:B1:56:E5:4D:C7:A6:B1:
  7D:0D:2F:B0:ED:39:AF:8C:C9:8D:29:3E:B9:17:82:6D:
  79:EC:9A:E4:C4:42:7B:18:13:9E:A2:06:C0:4B:A8:85:
  6E:67:92:A8:F1:E3:F3:E2:41:1F:9B:2D:24:D9:DF:4C:
  2B:A1:68:CE:96:C7:AF:F7:5B:F7:3D:2F:06:57:39:74:
  CF:B2:FA:46:C6:AD:18:60:8D:3E:0C:F7:C1:66:52:37:
  CF:89:42:B0:D7:33:C4:95:7E:F4:D9:1E:32:B8:5E:12:
```

第11章 TKSTOOL （トークンキーの管理）

TKS ユーティリティー **tkstool** は、トークンに保存されているキー、TKS マスターキー、および関連するキーおよびデータベースを含むキーを管理します。

11.1. SYNTAX

tkstool を使用して、さまざまな方法で証明書および鍵を管理できます。これらの異なる操作の構文は、以下のとおりです。

- トークンからのキーの削除

  tkstool -D -n *keyname* **-d** *dbdir* **[-h** *token_name***] [-p** *dbprefix***] [-f** *pwfile***]**

- ファイル共有を入力して、新しいトランスポートキーを生成します。

  tkstool -I -n *keyname* **-d** *dbdir* **[-h** *token_name***] [-p** *dbprefix***] [-f** *pwfile***]**

- 指定されたキーのキーチェック値(KCV)を表示します。

  tkstool -K -n *keyname* **-d** *dbdir* **[-h** *token_name***] [-p** *dbprefix***] [-f** *pwfile***]**

- 指定されたキーまたはすべてのキーの一覧表示。

  tkstool -L -n *keyname* **-d** *dbdir* **[-h all** | **-h** *token_name***]**
  [-p *dbprefix***] [-f** *pwfile***] [-x]**

- 新規マスターキーの生成。

  tkstool -M -n *keyname* **-d** *dbdir* **[-h** *token_name***] [-p** *dbprefix***] [-f** *pwfile***]**

- 新しいキーデータベースの作成。

  tkstool -N -d *dbdir* **[-p** *dbprefix***] [-f** *pwfile***]**

- キーデータベースのパスワードの変更。

  tkstool -P -d *dbdir* **[-p** *dbprefix***] [-f** *pwfile***]**

- 対称キーの名前を変更します。

  tkstool -R -n *keyname* **-r** *new_keyname* **-d** *dbdir* **[-h** *token_name***]**
  [-p *dbprefix***] [-f** *pwfile***]**

- すべてのセキュリティーモジュールの一覧表示。

  tkstool -S -d *dbdir* **[-p** *dbprefix***] [-x]**

- 新しいトランスポートキーを生成します。

```
tkstool -T -n keyname -d dbdir [-h token_name]
 [-p dbprefix] [-f pwfile] [-z noiseFile]
```

- ラップされたマスターキーをアンラップします。

```
tkstool -U -n keyname -d dbdir -t transport_keyname -i inputFile
 [-h token_name] [-p dbprefix] [-f pwfile]
```

- 新規マスターキーをラップします。

```
tkstool -W -n keyname -d dbdir -t transport_keyname -o outputFile
 [-h token_name] [-p dbprefix] [-f pwfile]
```

注記

tkstool の -R オプションのバージョン 1.0 をサポートするには、Chrysalis-ITS バージョン 2.3 が必要です。

以前のバージョンの tkstool によって作成された Chrysalis-ITS ハードウェアトークンに存在するトランスポートキーは、CKA_ENCRYPT ビットおよび CKF_ENCRYPT ビットが以前のツールで作成されるときに設定されないため、KCV 値を tkstool の -K オプションで決定できません。

tkstool オプションは以下のとおりです。

オプション	説明
D	トークンからキーを削除します。
d	必須。セキュリティーモジュールデータベース（その操作で許可されている場合は HSM）またはキーデータベースディレクトリー（ソフトウェア）を提供します。
f	パスワードファイルのパスとファイル名を指定します（ある場合）。
h	管理するキーが含まれる toke のトークン名を指定します。一部の操作では、すべて のオプションがトークンのすべてのキーを管理できます。
l	新しいトランスポートキーを生成するための入力共有。
i	-U で必須ラップされたマスターキーを含む入力ファイルのパスおよびファイル名を指定します。
K	指定されたキーの KCV を表示します。

オプション	説明
L	指定されたキーまたはすべてのキーを一覧表示します。
M	新しいマスターキーを生成します。
N	新しいキーデータベース（ソフトウェア）を作成します。
n	-N、-P、および -S 以外のすべての操作に必要です。管理対象キーの名前を指定します。
o	-W で必須新しいラップされたマスターキーを出力するファイルのパスおよびファイル名を指定します。
P	キーデータベースのパスワード（ソフトウェア）を変更します。
p	キーデータベースディレクトリーに接頭辞を指定します。
R	対称鍵の名前を変更します。
r	-R で必須新しいキー名を指定します。
S	すべてのセキュリティーモジュールを一覧表示します。
T	新しいトランスポートキーを生成します。
t	-U および -W で必須管理対象トランスポートキーの名前を指定します。
U	ラップされたマスターキーをアンラップします。
W	新規マスターキーをラップします。
x	データベースの読み取り/書き込みを強制します。
z	キーを生成するためにノイズファイルのパスとファイル名を指定します。

ユーティリティーに関する詳細情報を取得するために **tkstool** で使用できる追加オプションは 2 つあります。

オプション	説明
H	拡張ヘルプ情報を表示します。
V	tkstool ツールのバージョン番号を表示します。

11.2. 使用方法

1. 次のコマンドを実行して、tkstool のバージョンを確認します。

```
tkstool -V
```

このコマンドを実行すると、以下のような出力が返されます。

```
tkstool: Version 1.0
```

2. 新しいソフトウェアデータベースを作成します。

```
tkstool -N -d .
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
Re-enter password:
```

注記

modutil ユーティリティーが最初に HSM スロットとトークンを secmod.db データベースに挿入するために使用される場合は、ハードウェア HSM をソフトウェアデータベースの代わりに使用できます。

HSM を使用する場合は、以下のコマンドにオプション -h hsm_token を追加する必要があります。

3. ローカルソフトウェアキーデータベースの内容を一覧表示します。

```
tkstool -L -d .

slot: NSS User Private Key and Certificate Services
token: NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
tkstool: the specified token is empty
```

4. transport という トランスポート キーを作成します。

```
tkstool -T -d . -n transport
```

5. プロンプトが表示されたら、データベースのパスワードを入力し、noise と入力して乱数ジェネレーターを確認します。

6. セッションキー共有と対応する KCV が表示されます。これら両方を書き留めておきます。

7. 以下のコマンドを実行して、同一のトランスポートキーを生成します。これは通常、同一のトランスポートキーを使用する必要がある別のデータベースセット内で使用されます。これが実行されると、複数のセッションキー共有と KCV が生成されます。この情報をすべて書き留めておきます。

```
tkstool -I -d . -n verify_transport
```

以下のような応答が表示されます。

```
Generating first symmetric key . . .
Generating second symmetric key . . .
Generating third symmetric key . . .
Extracting transport key from operational token . . .
    transport key KCV: A428 53BA
Storing transport key on final specified token . . .
Naming transport key "transport" . . .
Successfully generated, stored, and named the transport key!
```

8. キーデータベースの内容を再度一覧表示します。

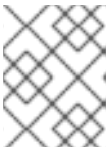
```
tkstool -L -d .

 slot: NSS User Private Key and Certificate Services
token: NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
 0 transport
```

9. トランスポートキーを使用してマスターキーを生成してラップし、マスターキーをファイル と呼ばれる ファイル に保存します。

```
tkstool -W -d . -n wrapped_master -t transport -o file

Enter Password or Pin for "NSS Certificate DB":
Retrieving the transport key (for wrapping) from the specified token . . .
Generating and storing the master key on the specified token . . .
Naming the master key "wrapped_master" . . .
Successfully generated, stored, and named the master key!
Using the transport key to wrap and store the master key . . .
Writing the wrapped data (and resident master key KCV) into the file
called "file" . . .

    wrapped data:   47C0 06DB 7D3F D9ED
                    FE91 7E6F A7E5 91B9
    master key KCV: CED9 4A7B
    (computed KCV of the master key residing inside the wrapped data)
```

10. ソフトウェアキーデータベースの内容を再度一覧表示します。

```
tkstool -L -d .

 slot: NSS User Private Key and Certificate Services
token: NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
 0 wrapped_master
 1 transport
```



注記

キーの順序は重要ではなく、一部のシステムでは別の順序でキーが表示される場合があります。

11. **トランスポートキーを使用して、ファイル と呼ばれるファイルに保存される unwrapped_master というマスターキーを生成およびアンラップします。**

```
tkstool -U -d . -n unwrapped_master -t transport -i file

Enter Password or Pin for "NSS Certificate DB":
Retrieving the transport key from the specified token (for unwrapping) . . .
Reading in the wrapped data (and resident master key KCV) from the file
called "file" . . .

    wrapped data:   47C0 06DB 7D3F D9ED
                    FE91 7E6F A7E5 91B9
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to temporarily unwrap the master key to
recompute its KCV value to check against its pre-computed KCV value . . .
    master key KCV: CED9 4A7B
    (computed KCV of the master key residing inside the wrapped data)
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to unwrap and store the master key on the
specified token . . .
Naming the master key "unwrapped_master" . . .
Successfully unwrapped, stored, and named the master key!
```

12. **キーデータベースの内容を一覧表示し、すべてのキーを表示します。**

```
tkstool -L -d .

 slot: NSS User Private Key and Certificate Services
token: NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
 0 unwrapped_master
 1 wrapped_master
 2 transport
```

13. **データベースからキーを削除します。**

```
tkstool -D -d . -n wrapped_master

Enter Password or Pin for "NSS Certificate DB":
tkstool: 1 key(s) called "wrapped_master" were deleted
```

14. **キーデータベースの内容を再度一覧表示して、すべてのキーを表示します。**

```
tkstool -L -d .

slot: NSS User Private Key and Certificate Services
token: NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
 0 unwrapped_master
 1 transport
```

第12章 CMCREQUEST (CMC 要求の作成)

CMC Request ユーティリティー **CMCRequest** は、1つ以上の PKCS #10 または CRMF 要求から CMC 要求を作成します。ユーティリティーは、証明書の取り消しにも使用できます。

12.1. SYNTAX

CMCRequest コマンドは、設定ファイル（）をパラメーターと して使用します。**.cfg** ファイルには、フォーマットされた CMC 要求のファイルへのパスを含める必要があります。

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

失効要求の場合、**revRequest.enable** パラメーターを **true** に設定し、関連パラメーターに適切な情報が含まれている必要があります。

.cfg ファイルには以下のパラメーターが含まれます。

パラメーター	説明
numRequests	PKCS #10 または CRMF 要求の合計数。このパラメーターの値は 0 になる場合があります。 たとえば、**numRequests=1** です。
入力 (input)	PKCS #10 または CRMF 要求の完全パスおよびファイル名。base-64 でエンコードされた形式である必要があります。複数のファイル名は空白で区切られます。**numRequests** の値が 0 よりも大きい場合は、このパラメーターは必須です。 たとえば、**input=crmf1** です。
出力 (output)	必須。生成されたバイナリー CMC 要求の完全パスおよびファイル名。 たとえば、**output=cmc** です。
ニックネーム	必須。完全な CMC 要求の署名に使用されるエージェント証明書のニックネーム。 たとえば、**nickname=CS Agent-102504a** の **102504a ID** です。
dbdir	必須。**cert8.db**、**key3.db**、および **secmod.db** データベースがあるディレクトリーへの完全パス。これは通常、ホームディレクトリーのブラウザー証明書データベースなどのエージェントの個人ディレクトリーです。 例： **~jsmith/.mozilla/firefox**
password	必須。エージェント証明書を格納 する **cert8.db** のトークンパスワード。 たとえば、**password=secret** です。

パラメーター	説明
format	要求形式(pkcs10 または crmf のいずれか)。 たとえば、format=crmf です。

以下の .cfg ファイルパラメーターは、CMC コントロールを設定します。

パラメーター	説明
confirmCertAcceptance.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、confirmCertAcceptance.enable=false です。
confirmCertAcceptance.serial	confirmCertAcceptance 制御のシリアル番号。 たとえば、confirmCertAcceptance.serial=3 です。
confirmCertAcceptance.issuer	confirmCertAcceptance 制御の発行者名。 たとえば、confirmCertAcceptance.issuer=cn=Certificate Manager,ou=102504a,o=102504a,c=us などです。
getCert.enable	true に設定すると、リクエストにはこの属性が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、getCert.enable=false です。
getCert.serial	getCert 制御のシリアル番号。 たとえば、getCert.serial=300 です。
getCert.issuer	getCert 制御の発行者名。 たとえば、getCert.issuer=cn=Certificate Manager,ou=102504a,o=102504a,c=us などです。
dataReturn.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、dataReturn.enable=false です。
dataReturn.data	dataReturn コントロールに含まれるデータ。 たとえば、dataReturn.data=test です。

パラメーター	説明
transactionMgt.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、transactionMgt.enable=true です。
transactionMgt.id	transactionMgt コントロールのトランザクション識別子。Verisign は、トランザクション ID を公開鍵の MD5 ハッシュにすることを推奨しています。
senderNonce.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、senderNonce.enable=false です。
senderNonce.id	senderNonce 制御の ID。 たとえば、senderNonce.id=testing です。
revRequest.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、revRequest.enable=true です。
revRequest.nickname	取り消される証明書のニックネーム。 たとえば、revRequest.nickname=newuser の 102504a ID です。
revRequest.issuer	失効する証明書の発行者名。 たとえば、revRequest.issuer=cn=Certificate Manager,ou=102504a,o=102504a,c=us などです。
revRequest.serial	失効する証明書のシリアル番号。 たとえば、revRequest.serial=75 です。
revRequest.reason	この証明書を取り消す理由。許可される値は、指定されていない、keyCompromise、caCompromise、affiliationChanged で、代わりに、cessationOfOperation、certificateHold、および removeFromCRL に置き換えられ ます。 たとえば、revRequest.reason=unspecified です。
revRequest.sharedSecret	失効要求の共有シークレット。 たとえば、revRequest.sharedSecret=testing です。

パラメーター	説明
revRequest.comment	失効要求のテキストコメント。 たとえば、revRequest.comment=readable コメント です。
revRequest.invalidityDatePresent	true に設定すると、現在の時間は失効した証明書の無効化日になります。false に設定すると、無効な日付は存在しません。 たとえば、revRequest.invalidityDatePresent=false です。
identityProof.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、identityProof.enable=false です。
identityProof.sharedSecret	identityProof 制御の共有シークレット。 たとえば、identityProof.sharedSecret=testing です。
popLinkWitness.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 例： popLinkWitness.enable=false
LraPopWitness.enable	true に設定すると、リクエストにはこの制御が含まれます。このパラメーターが設定されていない場合、値は false であると想定されます。 たとえば、LraPopWitness.enable=false です。
LraPopWitness.bodyPartIDs	LraPopWtiness 制御の本文部分 ID のスペース区切りリスト。 たとえば、LraPopWitness.bodyPartIDs=1 です。

12.2. 使用方法

PKCS#10 要求を含む単純な CMC 要求が生成されたら、それを CA に送信します。最も簡単な方法は、エンドエンティティーページを使用することです。

1. エンドエンティティーサービスページを開きます。

   https://server.example.com:9444/ca/ee/ca/

2. caCMCUserCert プロファイルを選択します。

3. CMC 要求に貼り付けます。

または、HttpClient を使用してプロファイルに投稿します。

1. AtoB ツールを実行して、base-64 でエンコードされた PKCS #10 リクエストをバイナリーに変換します。
2. HttpClient ユーティリティーを使用して要求を送信します。

   /ca/ee/ca/profileSubmitCMCFull および /ca/ee/ca/profileSubmitCMCSimple など、CMC 要求を送信できるプロファイルは複数あります。プロファイルは HttpClient 設定で指定する必要があります。

12.3. 出力

CMCRequest コマンドは、.cfg ファイルのパラメーターに応じて証明書要求を生成します。例 12.1「CMC 要求 .cfg ファイル」 のパラメーターは、例12.2「CMC 要求の出力」 でリクエストを作成するために使用されます。

例12.1 CMC 要求 .cfg ファイル

```
#Usage: CMCRequest <configuration file>
#For example, CMCRequest CMCRequest.cfg

#The configuration file should look like as follows:

#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=pkcs10.i
#input=govReq2.txt
input=myCMC.txt

#output: full path for the CMC request in binary format
output=/tmp/cfu/cmcReq.myCMC

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
#nickname=CMS Agent Certificate
#nickname=cfuAgent-ca2's SjcRedhat Domain jaw ca2 ID
nickname=CA Administrator of Instance pki-ca-0124's SjcRedhat Domain 0124 ID

#dbdir: directory for cert8.db, key3.db and secmod.db
dbdir=/tmp/cfu/

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

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

#confirmCertAcceptance.enable: if true, then the request will
#contain this control. Otherwise, false.
```

```
confirmCertAcceptance.enable=false

#confirmCertAcceptance.serial: The serial number for
#confirmCertAcceptance control
confirmCertAcceptance.serial=3

#confirmCertAcceptance.issuer: The issuer name for
#confirmCertAcceptance control
confirmCertAcceptance.issuer=cn=Certificate Manager,c=us

#getCert.enable: if true, then the request will contain this
#control. Otherwise, false.
getCert.enable=false

#getCert.serial: The serial number for getCert control
getCert.serial=3

#getCert.issuer: The issuer name for getCert control
getCert.issuer=cn=Certificate Manager,c=us

#dataReturn.enable: if true, then the request will contain
#this control. Otherwise, false.
dataReturn.enable=false

#dataReturn.data: data contained in the control.
dataReturn.data=test

#transactionMgt.enable: if true, then the request will contain
#this control. Otherwise, false.
transactionMgt.enable=false

#transactionMgt.id: transaction identifier. Verisign recommend
#transactionId to be MD5 hash of publicKey.
transactionMgt.id=

#senderNonce.enable: if true, then the request will contain this
#control. Otherwise, false.
senderNonce.enable=false

#senderNonce.id: sender nonce
senderNonce.id=

#revRequest.enable: if true, then the request will contain this
#control. Otherwise, false.
revRequest.enable=false

#revRequest.nickname: The nickname for the revoke certificate
revRequest.nickname=newuser's 102504a ID

#revRequest.issuer: The issuer name for the certificate being
#revoked.
revRequest.issuer=cn=Certificate Manager,c=us

#revRequest.serial: The serial number for the certificate being
#revoked.
revRequest.serial=61
```

```
#revRequest.reason: The reason for revoking this certificate:
#               unspecified, keyCompromise, caCompromise,
#               affiliationChanged, superseded, cessationOfOperation,
#               certificateHold, removeFromCRL
revRequest.reason=unspecified

#revRequest.sharedSecret: The sharedSecret
revRequest.sharedSecret=

#revRequest.comment: The human readable comment
revRequest.comment=

#revRequest.invalidityDatePresent: if true, the current time will be the
#                       invalidityDate. If false, no invalidityDate
#                       is present.
revRequest.invalidityDatePresent=false

#identityProof.enable: if true, then the request will contain
#this control. Otherwise, false.
identityProof.enable=false

#identityProof.sharedSecret: Shared Secret
identityProof.sharedSecret=testing

#popLinkWitness.enable:  if true, then the request will contain
#this control. Otherwise, false.
#If you want to test this control, make sure to use CRMFPopClient
# to generate the CRMF request which will include the
#idPOPLinkWitness attribute in the controls section of the
#CertRequest structure.
popLinkWitness.enable=false

#LraPopWitness.enable: if true, then the request will contain this
#control. Otherwise, false.
LraPopWitness.enable=false

#LraPopWitness.bodyPartIDs: List of body part IDs
#Each id is separated by space.
LraPopWitness.bodyPartIDs=1
```

例12.2 CMC 要求の出力

```
CMCRequest CMCrequest.myCMC.cfg

cert/key prefix =
path = /tmp/cfu/

The CMC enrollment request in base-64 encoded format:

MIIKZwYJKoZIhvcNAQcCoIIKWDCCClQCAQMxCzAJBgUrDgMCGgUAMIIBxAYIKwYB
BQUHDAKgggG2BIIBsjCCAa4wADCCAaShggGgMIIBBgIFAPgzSl8wgceAAQKIDjAM
MQowCAYDVQQDEwF4poGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDhZcSEFI3v
YqNWHsHIH/BDrcVHLuHNuifuSE0fgyirNAwI7IwVReB/I2b1NWSyqh2+9PYIFeSc
```

```
VjXvh7p9GU7GmLL4p+Tdpx3YD1JVrumbn6W2uGvMf8UgNx8OxFgkuKy3Z9ohd30x
oTi/hEKoDKxUXN6BY93UPwKLQ7Fpo9RDvQIDAQABqRAwDgYDVR0PAQH/BAQDAgXg
MDMwFQYJKwYBBQUHBQEBDAhyZWdUb2tlbjAaBgkrBgEFBQcFAQIMDWF1dGhlbnRp
Y2F0b3KhgZMwDQYJKoZIhvcNAQEFBQADgYEAtewF4jFndWjpduAzxsxYmBGsPtrE
drCtsm7lvf1ytUPRX0dlEhKglEQBNsr/UZaCGWrCNpqdKjlSlbsZAw/0Jd8oiRYP
pd6sjYJmBoP5uCf/xft2tJAFDGBAeb3T4VwZb//SasrrRvl6Aa5PBqbh1FrjSCeo
Cc/VeX2nHgwKjj8wADAAolIHODCCA2owggJSoAMCAQICAQYwDQYJKoZIhvcNAQEL
BQAwUTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZw
a2ktY2ExHjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1dGhvcml0eTAeFw0xMTAxMjQy
MzU3MTVaFw0xMzAxMTMyMzU3MTVaMIGJMR4wHAYDVQQKExVTamNSZWRoYXQgRG9t
YWluIDAxMjQxHTAbBgkqhkiG9w0BCQEWDmNmdUByZWRoYXQuY29tMRUwEwYKCZIm
iZPyLGQBARMFYWRtaW4xMTAvBgNVBAMTKENBIEFkbWluaXN0cmF0b3Igb2YgSW5z
dGFuY2UgcGtpLWNhLTAxMjQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANGu
Qk6xUMkuY8j1/NxXBBEz0N1zZgziqGDMLmQorYxVkIDsCMx9tajq3/r9u2CDLal0
QTvbUwPd1V+CDPfopHG1eTOL62bzLdF1874Q8OW0+UD9m6IFYgnY0toqJJLU/1eO
JUPkbYnGJwmfG3MTWbpr2MrEr+wwalPgmytlaOzxAgMBAAGjgZcwgZQwHwYDVR0j
BBgwFoAU10BlukYi0n1jHqDlvwut/A0qdHswQgYIKwYBBQUHAQEENjA0MDIGCCsG
AQUFBzABhiZodHRwOi8vcGF3LnNqYy5yZWRoYXQuY29tOjkxODAvY2Evb2NzcDAO
BgNVHQ8BAf8EBAMCBPAwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMA0G
G
CSqGSIb3DQEBCwUAA4IBAQCwQEmjjVmmgEdAO/EYaTQXmfRhEsMYuDium6EoKCpC
Qb4JReUXekxrJnTpTwkUbJq6xiuDozrLHryWAnk1Y6WHxILUkJppCvCiXcVkicvV
eGU2S6p8hKPbC5LLThotN1OlU74N8fdE+zunFV+xnP/4GkJQKuNJiRTZOFmvh/jY
QlqDBcNPhVfcu200H1UaHqLxG22gEByxqs/ma13MEQtaMZBAvicc4i5vhT01YwT2
suYcJDmYpaWVKTjXtm572lNgMYMpNjxnRowicq5Ez8oj5CZc39fB3l3u8fBCRzqo
PlDVQZFzNP+xyvzyJRhUc5oeglaealOdh28X9OXe+eE8MIIDxjCCAq6gAwIBAgIB
ATANBgkqhkiG9w0BAQsFADBRMR4wHAYDVQQKExVTamNSZWRoYXQgRG9tYWluIDAx
MjQxDzANBgNVBAsTBnBraS1jYTEeMBwGA1UEAxMVQ2VydGlmaWNhdGUgQXV0aG9y
aXR5MB4XDTExMDEyNDIzNTYxMloXDTE5MDEyNDIzNTYxMlowUTEeMBwGA1UEChMV
U2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZwa2ktY2ExHjAcBgNVBAMT
FUNlcnRpZmljYXRlIEF1dGhvcml0eTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBANIRZ/b8FFn/8FgVXXg4scSuzTgZ61/upie2zt0n/hY2eMRYh12tlocX
k64WYEREvKAFLF8pYMfoZzldylp9vEWChEWd8OqOM6pcKRpxnphNSOsTIAFh+QbD
rnvusCg63idr4WLiEP92dXZEpls1m0bCXnKOF2Vio0CX7VM8X2iHQVKOolQzovsl
Kc+xt/5p/Hy9vFDF+Lyf5dBnT3Rsct/T+Z1pNnHeS5bnv28oxXRdSnnrPPEEVDq2
jj+k1hje4b1alVuEyGgcKWrlnyZXSei4nY0WDmEv/Lgox6o+QyVEmLMydWj8G5d0
XreQZYke9+XS6OFNah8fFVLW+GCeqtkCAwEAAaOBqDCBpTAfBgNVHSMEGDAWgBTX
QGW6RiLSfWMeoMi/C638DSp0ezAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQE
AwIBxjAdBgNVHQ4EFgQU10BlukYi0n1jHqDlvwut/A0qdHswQgYIKwYBBQUHAQEE
NjA0MDIGCCsGAQUFBzABhiZodHRwOi8vcGF3LnNqYy5yZWRoYXQuY29tOjkxODAv
Y2Evb2NzcDANBgkqhkiG9w0BAQsFAAOCAQEAEfEaydNlzEO6cUEnw9Q3aLf5UcRQ
/K+wggfvtBN33moQD6Z6MmOGiQh/s2bgwDtYgoCnwhkLlpQggZZ2R/Q4b7LV5tzH
B1+v40LZsC4bQ6BPkUlX5gzoCZNJiNlM4Bc+tg92MWlYKj5zHr6yghiJATr87vBY
UxeUOTH7d5i9X6TlCsf8AEb50WMFPaoW9GctTwelVYlgg56dFC3wY81bdEBr0SlD
l1lW97WuoPU+Jh1OA0AANcYlOh5j9fyOlsqcdUXhPQUsTq2Ou20jpOrh0Aw6CHpQ
3S4rYJSg7MEbl3lQFOapAfOqrl1e3kfgogolIEQmhOOrjpUnQc+9C7l/gDGCATww
ggE4AgEDMFYwUTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYD
VQQLEwZwa2ktY2ExHjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1dGhvcml0eQIBBjAJ
BgUrDgMCGgUAoD4wFwYJKoZIhvcNAQkDMQoGCCsGAQUFBwwCMCMGCSqGSIb3DQEJ
BDEWBBTJWrAxeErsabiWVokJhrYe8O2AXDANBgkqhkiG9w0BAQEFAASBgJSrhYMo
smKomXTGaczljvhYj7lsCUgbpPMqzfhQh5l1X2b5hL3hkWaMDDl9eo2HGZYoE9Lr
6RolMNs8FCN8F6F8eBzRKlkZTEA+3nXB7gnYVbxrwJrlm2htyTgphu6/yck0wCH9
Og2BekSHQsJ7V7abP04U0VBIUAocJmHwILnQ
```

The CMC enrollment request in binary format is stored in /tmp/cfu/cmcReq.myCMC.

第13章 CMCENROLL (CMC 登録の実行)

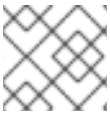
CMC 登録ユーティリティー CMCEnroll は、エージェントの証明書を使用して証明書要求に署名するために使用されます。これは、CA エンドエンティティー CMC 登録 フォームと組み合わせて使用して、ユーザーの証明書に署名および登録できます。

13.1. SYNTAX

このユーティリティーの構文は以下のとおりです。

```
CMCEnroll -d directory_containing_agent_cert -n certificate_nickname -r certificate_request_file -p certificate_DB_passwd [-c comment]
```

オプション	説明
d	エージェント証明書に関連付けられた **cert8.db ファイル、key3.** db ファイル、および **secmod.db** ファイルを含むディレクトリー。これは通常、ホームディレクトリーのブラウザー証明書データベースなどのエージェントの個人ディレクトリーです。
n	要求の署名に使用されるエージェント証明書のニックネーム。
r	証明書要求のファイル名。
p	**-d** で指定されたエージェント証明書が含まれる NSS 証明書データベースのパスワード。



注記

引用符で囲まれたスペースを含む値を囲みます。

13.2. 使用方法

署名付き要求は、処理される CA に提出する必要があります。

1. **certutil などのツールを使用して、PKCS #10 証明書要求を作成します。**
2. **PKCS #10 ASCII 出力をテキストファイルにコピーします。**
3. **CMCEnroll コマンドを実行して、証明書要求に署名します。入力ファイルが request34.txt の場合、エージェントの証明書は ~jsmith/.mozilla/firefox ディレクトリーに保存され、この CA の証明書の共通名は Certificate Manager Agents Cert で、証明書データベースのパスワードは 1234pass です。**

```
CMCEnroll -d "~jsmith/.mozilla/firefox" -n "Certificate Manager Agents Cert" -r "/export/requests/request34.txt" -p "1234pass"
```

このコマンドの出力は、同じファイル名で、ファイル名に .out が追加されたファイルに保存されます。

4. CA エンドエンティティーページから署名済み証明書を送信します。

   a. エンドエンティティーを開きます。

   b. CMC 登録プロファイルフォームを選択します。

   c. 出力ファイルの内容をこのフォームの最初のテキスト領域に貼り付けます。

   d. 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。

   e. Certificate Type User Certificate を選択し、連絡先情報を入力してフォームを送信します。

5. 署名付き要求が送信され、CMCAuth プラグインが有効になってから、証明書は即座に処理され、返されます。

6. エージェントページを使用して、新しい証明書を検索します。

13.3. 出力

CMCEnroll は、標準の PKCS #10 証明書要求を取得し、エージェント証明書を使用して署名します。出力は、適切なプロファイルから送信できる証明書要求でもあります。

```
# CMCEnroll -d ~jsmith/.mozilla/firefox -n "CA Administrator of Instance pki-ca Example Domain ID" -
r pkcs10.i -p secret
cert/key prefix =
path = .
-----BEGIN NEW CERTIFICATE REQUEST-----
MIILsQYJKoZIhvcNAQcCoIILojCCC54CAQMxCzAJBgUrDgMCGgUAMIIDDgYIKwYB
BQUHDAKgggMABIIC/DCCAvgwUzAtAgECBggrBgEFBQcHBjEeBBxlWXlObThoN1dk
eDNHcEE1MllxcERTNXVRVlE9MCICAQMGCCsGAQUFBwcFMRMCEQD6Yglp1fV/P0VA
CzSQPuO9MIICm6CCApcCAQEwggKQMIIBeAIBADBLMSUwIwYDVQQKExxTZmJheSBS
ZWRoYXQgRG9tYWluIHBhdyAxMjEyMSIwIAYDVQQDExlEUk0gU3Vic3lzdGVtIENl
cnRpZmljYXRlMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqZvFwMGg
f2W0/OAlPBLcCq5dlHkrrQIsKx4P2c1Q+8eFJAxOPLaA5xVs+AGHMUGeBTsRuk0Q
26NRWq9JOtrRbIaUFy4ilTI7vpM2NThObMidE8EUs8p4wef5C0kTgkKA1/B+F6IF
JU8LVMtF2hHOTUEPheRMyb4fTJPjUec+zNduKe2xdlrJmmUgCC8ibC4KavyKDe99
T89CcXNZe/NgyejILhZ95wPzaW7xmN36fXUrxsMDKLJSn4X2i5DlKyuybw4+rosv
jI2J9o2Kfj1EDHHkzC57iYjNF4wmgUyZobnSYNyNLZWzbXiGEHq8/yWaxZxML4fP
SYspU2EIWh0L/QIDAQABoAAwDQYJKoZIhvcNAQEEBQADggEBAEJWi4aUqFJ4uEi3
xwf9bAdPhRpifdfWZN1BfVQn5DoaI3Bhm4EDsVuugm7E4IIls1SysmsQhk2F5A3p
ltpGCnoMXnFg9ToCpCu6wKi9xRwT5+2Iwbjnj1JtMkQx90xebBQ9v3+VVClRXjJs
edp3OSDIB/LA5/ieg+be/VbWaTb1yrvGaZruZT+iu2GPGQ8R40uPM3n22K/Vu3LI
dhO5uNIP8zAaEpda16detBXievCZgZsuC8iRkKfrRir97L2zJk46/xMgno6CqQFL
4p9gH5/G/vddH2pbNI8kqoIEthmSw6mD/yQjaSAsAC8DM8wWAo4SIz6MXTcMPGgO
moDuVVowADAAoIIHODCCA2owggJSoAMCAQICAQYwDQYJKoZIhvcNAQELBQAwUTEe
MBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZwa2ktY2Ex
HjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1dGhvcml0eTAeFw0xMTAxMjQyMzU3MTVa
Fw0xMzAxMTMyMzU3MTVaMIGJMR4wHAYDVQQKExVTamNSZWRoYXQgRG9tYWluIDAx
MjQxHTAbBgkqhkiG9w0BCQEWDmNmdUByZWRoYXQuY29tMRUwEwYKCZImiZPyLGQB
ARMFYWRtaW4xMTAvBgNVBAMTKENBIEFkbWluaXN0cmF0b3Igb2YgSW5zdGFuY2Ug
cGtpLWNhLTAxMjQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANGuQk6xUMku
```

```
Y8j1/NxXBBEz0N1zZgziqGDMLmQorYxVklDsCMx9tajq3/r9u2CDLaI0QTvbUwPd
1V+CDPfopHG1eTOL62bzLdF1874Q8OW0+UD9m6IFYgnY0toqJJLU/1eOJUPkbYnG
JwmfG3MTWbpr2MrEr+wwaIPgmytlaOzxAgMBAAGjgZcwgZQwHwYDVR0jBBgwFoAU
10BlukYi0n1jHqDIvwut/A0qdHswQgYIKwYBBQUHAQEENjA0MDIGCCsGAQUFBzAB
hiZodHRwOi8vcGF3LnNqYy5yZWRoYXQuY29tOjkxODAvY2Evb2NzcDAOBgNVHQ8B
Af8EBAMCBPAwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMA0GCSqGSIb3
DQEBCwUAA4IBAQCwQEmjjVmmgEdAO/EYaTQXmfRhEsMYuDium6EoKCpCQb4JReUX
ekxrJnTpTwkUbJq6xiuDozrLHryWAnk1Y6WHxILUkJppCvCiXcVkicvVeGU2S6p8
hKPbC5LLThotN1OIU74N8fdE+zunFV+xnP/4GkJQKuNJiRTZOFmvh/jYQIqDBcNP
hVfcu200H1UaHqLxG22gEByxqs/ma13MEQtaMZBAvicc4i5vhT01YwT2suYcJDmY
paWVKTjXtm572lNgMYMpNjxnRowicq5Ez8oj5CZc39fB3l3u8fBCRzqoPlDVQZFz
NP+xyvzyJRhUc5oegIaealOdh28X9OXe+eE8MIIDxjCCAq6gAwIBAgIBATANBgkq
hkiG9w0BAQsFADBRMR4wHAYDVQQKExVTamNSZWRoYXQgRG9tYWluIDAxMjQxDzAN
BgNVBAsTBnBraS1jYTEeMBwGA1UEAxMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4X
DTExMDEyNDIzNTYxMloXDTE5MDEyNDIzNTYxMlowUTEeMBwGA1UEChMVU2pjUmVk
aGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZwa2ktY2ExHjAcBgNVBAMTFUNlcnRp
ZmljYXRlIEF1dGhvcml0eTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
ANlRZ/b8FFn/8FgVXXg4scSuzTgZ61/upie2zt0n/hY2eMRYh12tlocXk64WYERE
vKAFLF8pYMfoZzldylp9vEWChEWd8OqOM6pcKRpxnphNSOsTlAFh+QbDrnvusCg6
3idr4WLiEP92dXZEpIs1m0bCXnKOF2Vio0CX7VM8X2iHQVKOoIQzovslKc+xt/5p
/Hy9vFDF+Lyf5dBnT3Rsct/T+Z1pNnHeS5bnv28oxXRdSnnrPPEEVDq2jj+k1hje
4b1aIVuEyGgcKWrlnyZXSei4nY0WDmEv/Lgox6o+QyVEmLMydWj8G5d0XreQZYke
9+XS6OFNah8fFVLW+GCeqtkCAwEAAaOBqDCBpTAfBgNVHSMEGDAWgBTXQGW6RiLS
fWMeoMi/C638DSp0ezAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQEAwIBxjAd
BgNVHQ4EFgQU10BlukYi0n1jHqDIvwut/A0qdHswQgYIKwYBBQUHAQEENjA0MDIG
CCsGAQUFBzABhiZodHRwOi8vcGF3LnNqYy5yZWRoYXQuY29tOjkxODAvY2Evb2Nz
cDANBgkqhkiG9w0BAQsFAAOCAQEAEfEaydNIzEO6cUEnw9Q3aLf5UcRQ/K+wggfv
tBN33moQD6Z6MmOGiQh/s2bgwDtYgoCnwhkLlpQggZZ2R/Q4b7LV5tzHB1+v40LZ
sC4bQ6BPkUIX5gzoCZNJiNlM4Bc+tg92MWIYKj5zHr6yghiJATr87vBYUxeUOTH7
d5i9X6TICsf8AEb50WMFPaoW9GctTwelVYlgg56dFC3wY81bdEBr0SIDl1lW97Wu
oPU+Jh1OA0AANcYlOh5j9fyOlsqcdUXhPQUsTq2Ou20jpOrh0Aw6CHpQ3S4rYJSg
7MEbl3lQFOapAfOqrl1e3kfgogoIIEQmhOOrjpUnQc+9C7l/gDGCATwwggE4AgED
MFYwUTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZw
a2ktY2ExHjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1dGhvcml0eQIBBjAJBgUrDgMC
GgUAoD4wFwYJKoZIhvcNAQkDMQoGCCsGAQUFBwwCMCMGCSqGSIb3DQEJBDEWBBQa
be8HVhF1o9FP9E2tjfOMkdmuQDANBgkqhkiG9w0BAQEFAASBgDAXIZgfFWv9GcLd
+yRET7HOgi1uTAiWwRoV2KZ0hbDowuNsaMHu5k4oic8wcOCOyAvWS/9o64RPG96f
QKLHrAa8CJ3hqzDl3xaOuNF/iJWM4R47136DRhW9cCA1qFH3FyiUcwBdGd5R8DrE
r+ce2ZSTtI2Jpif83w7Ro5VqSMMN
-----END NEW CERTIFICATE REQUEST-----
```

第14章 CMCRESPONSE (CMC 応答のプルーニング)

CMC Response ユーティリティー **CMCResponse** は、ユーティリティーが受信した CMC 応答を解析します。

14.1. SYNTAX

CMC Response ユーティリティーは、以下の構文を使用します。

> **CMCResponse -d** *directoryName* **-i** */path/to/CMCResponse.file*

Options	説明
d	**cert8.db** ディレクトリーへのパスを指定します。これは通常、ホームディレクトリーのブラウザー証明書データベースなどのエージェントの個人ディレクトリーです。
i	CMC 応答ファイルのパスとファイル名を指定します。

解析された出力は画面に出力されます。

14.2. 使用方法および出力

CMCResponse の目的は、CMC 応答を解析することです。12章*CMCRequest (CMC 要求の作成)*で説明されているように、CMC 要求が生成され、CMC 形式で応答を返す CMC プロファイルに送信されます。一般的なユースケースの 1 つとして、**HttpClient** などのツールを使用してリクエストを送信し、応答を取得します。その応答は **CMCResponse** に送信され、解析されます。

最初のステップでは、**HttpClient** がリクエストを送信するために使用する **.cfg** ファイルを作成します。

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

#port: port number
port=9444

#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=/tmp/cfu/cmcReq.myCMC

#output: full path for the response in binary format
output=/tmp/cfu/cmcResponse.myCMC

#dbdir: directory for cert8.db, key3.db and secmod.db
#This parameter will be ignored if secure=false
dbdir=/tmp/cfu

#clientmode: true for client authentication, false for no client authentication
```

```
#This parameter will be ignored if secure=false
clientmode=false

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

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

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitCMCFull
```

その設定ファイルは HttpClient に渡され、バイナリー CMC 応答を受け取ります。

```
# HttpClient HttpClient.cfg

Total number of bytes read = 2667
handshake happened
Total number of bytes read = 2287
MIII6wYJKoZIhvcNAQcCoIII3DCCCNgCAQMxDjAMBghghkgBZQMEAQUAMDUGCCsG
AQUFBwwDoCkEJzAlMB8wHQIBAQYIKwYBBQUHBwExDjAMAgEAMAcCBQD4M0pfMAAw
AKCCBrowggLsMIIB1KADAgECAgEaMA0GCSqGSIb3DQEBCwUAMFExHjAcBgNVBAoT
FVNqY1JlZGhhdCBEb21haW4gMDEyNDEPMA0GA1UECxMGcGtpLWNhMR4wHAYDVQQD
ExVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMTEwMzA4MTY0MTMwWhcNMTEwOTA0
MTY0MTMwWjAMMQowCAYDVQQDEwF4MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDhZcSEFI3vYqNWHsHIH/BDrcVHLuHNuifuSE0fgyirNAwl7lwVReB/l2b1NWSy
qh2+9PYIFeScVjXvh7p9GU7GmLL4p+Tdpx3YD1JVrumbn6W2uGvMf8UgNx8OxFgk
uKy3Z9ohd30xoTi/hEKoDKxUXN6BY93UPwKLQ7Fpo9RDvQIDAQABo4GXMIGUMB8G
A1UdIwQYMBaAFNdAZbpGltJ9Yx6gyL8LrfwNKnR7MEIGCCsGAQUFBwEBBDYwNDAy
BggrBgEFBQcwAYYmaHR0cDovL3Bhdy5zamMucmVkaGF0LmNvbTo5MTgwL2NhL29j
c3AwDgYDVR0PAQH/BAQDAgXgMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcD
BDANBgkqhkiG9w0BAQsFAAOCAQEAQxdBWvoc5/0SKUGdWvhs4NPqU1cX4fjjUW8t
famLXyk37K7PZM/f4wlso37OuQUQO/tuGR0+8EoBD8NfFJwGcMLb1XlfR/2n/Ndq
TmT6qRnuCST4ucQBEtE8rYkFYZQ5Z22N8QPBjiNvoO5qs8X9xMzmbJrjSyNwGJHl
UBDLhyqgVLzdl80UycoFQPPp8vi4/+2/e1+FFRUjtGgNE1Yc5DdrTeST3h5nA/uS
htQRHj8fzSjE/07zEyMFc/IAmCV3xWkiQK2uHJBrYBKFYVEZ7YJQ6sO/q/IUdv3H
5x6YqEWMqqEJhxru6PRhHKU8WeECu+Z5O+wfla7BOCjz+AVvLDCCA8YwggKuoAMC
AQICAQEwDQYJKoZIhvcNAQELBQAwUTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFp
biAwMTI0MQ8wDQYDVQQLEwZwa2ktY2ExHjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1
dGhvcml0eTAeFw0xMTAxMjQyMzU2MTJaFw0xOTAxMjQyMzU2MTJaMFExHjAcBgNV
BAoTFVNqY1JlZGhhdCBEb21haW4gMDEyNDEPMA0GA1UECxMGcGtpLWNhMR4wHAYD
VQQDExVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDZUWf2/BRZ//BYFV14OLHErs04Getf7qYnts7dJ/4WNnjEWIdd
rZaHF5OuFmBERLygBSxfKWDH6Gc5XcpafbxFgoRFnfDqjjOqXCkacZ6YTUjrE5QB
YfkGw6577rAoOt4na+Fi4hD/dnV2RKSLNZtGwl5yjhdIYqNAI+1TPF9oh0FSjqCE
M6L7JSnPsbf+afx8vbxQxfi8n+XQZ090bHLf0/mdaTZx3kuW579vKMV0XUp56zzx
BFQ6to4/pNYY3uG9WiFbhMhoHClq5Z8mV0nouJ2NFg5hL/y4KMeqPkMIRJizMnVo
/BuXdF63kGWJHvfl0ujhTWofHxVS1vhgnqrZAgMBAAGjgagwgaUwHwYDVR0jBBgw
FoAU10BlukYi0n1jHqDIvwut/A0qdHswDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8B
Af8EBAMCAcYwHQYDVR0OBBYEFNdAZbpGltJ9Yx6gyL8LrfwNKnR7MEIGCCsGAQUF
BwEBBDYwNDAyBggrBgEFBQcwAYYmaHR0cDovL3Bhdy5zamMucmVkaGF0LmNvbTo5
MTgwL2NhL29jc3AwDQYJKoZIhvcNAQELBQADggEBABHxGsnTSMxDunFBJ8PUN2i3
+VHEUPyvsIIH77QTd95qEA+mejJjhoklf7Nm4MA7WIKAp8lZC5aUIIGWdkf0OG+y
1ebcxwdfr+NC2bAuG0OgT5FCF+YM6AmTSYjZTOAXPrYPdjFiGCo+cx6+solYiQE6
```

```
/O7wWFMXIDkx+3eYvV+kyArH/ABG+dFjBT2qFvRnLU8HpVWJYIOenRQt8GPNW3RA
a9EiA5dZVve1rqD1PiYdTgNAADXGJToeY/X8jpbKnHVF4T0FLE6tjrttl6Tq4dAM
Ogh6UN0uK2CUoOzBGyN5UBTmqQHzqq5dXt5H4KIKCCBEJoTjq46VJ0HPvQu5f4Ax
ggHMMIIByAIBAzBWMFExHjAcBgNVBAoTFVNqY1JlZGhhdCBEb21haW4gMDEyNDEP
MA0GA1UECxMGcGtpLWNhMR4wHAYDVQQDExVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkC
AQEwDAYIYIZIAWUDBAEFAKBKMBcGCSqGSIb3DQEJAzEKBggrBgEFBQcMAzAvBgkq
hkiG9w0BCQQxIgQgXUsQ5rl+G2aiKpAp68LLdF7uOcPDOYbWIacKxpwKfZIwDQYJ
KoZIhvcNAQEBBQAEggEAa4fQfye0ogzxpFYZd98JNZITuWeluDBv+HwZelaRWYn4
/YIbZyn98gBaX5V1NNXsmRO1D8iKa7O+4XORweFnEdzqLDQCzN/TFsnKqT8dYHQT
iY4kd2msBOqYa+x3ZKZoEGvRIPMCRXBMTKfSmq963NT7hCZyLA2jmATs4eYrNyQp
xHPzxrUy0Ftj/NJKNb6g3JtSinUp9RkNMArAyg0ORFCcRbCRQNmxYIFkTyE7/yVY
uaRyE7XlPoBqdo5BWgsQlD7GxK0PeSzTBoqmygLu7gZZfx7pghV4YrXliYtgMafA
GQwiK2Jj1zs/eRR3MN3TvhSYTzavNxq7MXGQVavLQQ==

The response in binary format is stored in /tmp/jsmith/cmcResponse.myCMC
```

HttpClient 応答の最後の部分は、CMC 応答ファイルがどこにあるかを示し、そのファイルは CMCResponse で使用できます。CMCResponse がファイルを解析すると、応答の pretty-print バージョンが表示されます。

```
# CMCResponse -d . -i cmcResponse.myCMC
Certificates:
   Certificate:
      Data:
         Version:  v3
         Serial Number: 0x1A
         Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
         Issuer: CN=Certificate Authority,OU=pki-ca,O=SjcRedhat Domain 0124
         Validity:
            Not Before: Tuesday, March 8, 2011 8:41:30 AM PST America/Los_Angeles
            Not  After: Sunday, September 4, 2011 9:41:30 AM PDT America/Los_Angeles
         Subject: CN=x
         Subject Public Key Info:
            Algorithm: RSA - 1.2.840.113549.1.1.1
            Public Key:
               Exponent: 65537
               Public Key Modulus: (1024 bits) :
                  E1:65:C4:84:14:8D:EF:62:A3:56:1E:C1:C8:1F:F0:43:
                  AD:C5:47:2E:E1:CD:BA:27:EE:48:4D:1F:83:28:AB:34:
                  0C:08:EC:8C:15:45:E0:7F:23:66:F5:35:64:B2:AA:1D:
                  BE:F4:F6:08:15:E4:9C:56:35:EF:87:BA:7D:19:4E:C6:
                  98:B2:F8:A7:E4:DD:A7:1D:D8:0F:52:55:AE:E9:9B:9F:
                  A5:B6:B8:6B:CC:7F:C5:20:37:1F:0E:C4:58:24:B8:AC:
                  B7:67:DA:21:77:7D:31:A1:38:BF:84:42:A8:0C:AC:54:
                  5C:DE:81:63:DD:D4:3F:02:8B:43:B1:69:A3:D4:43:BD
         Extensions:
            Identifier: Authority Key Identifier - 2.5.29.35
               Critical: no
               Key Identifier:
                  D7:40:65:BA:46:22:D2:7D:63:1E:A0:C8:BF:0B:AD:FC:
                  0D:2A:74:7B
            Identifier: 1.3.6.1.5.5.7.1.1
               Critical: no
               Value:
                  30:34:30:32:06:08:2B:06:01:05:05:07:30:01:86:26:
                  68:74:74:70:3A:2F:2F:70:61:77:2E:73:6A:63:2E:72:
```

```
            65:64:68:61:74:2E:63:6F:6D:3A:39:31:38:30:2F:63:
            61:2F:6F:63:73:70
        Identifier: Key Usage: - 2.5.29.15
          Critical: yes
          Key Usage:
            Digital Signature
            Non Repudiation
            Key Encipherment
        Identifier: Extended Key Usage: - 2.5.29.37
          Critical: no
          Extended Key Usage:
            1.3.6.1.5.5.7.3.2
            1.3.6.1.5.5.7.3.4
    Signature:
      Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Signature:
        43:17:41:5A:FA:1C:E7:FD:12:29:41:9D:5A:F8:6C:E0:
        D3:EA:53:57:17:E1:F8:E3:51:6F:2D:7D:A9:8B:5F:29:
        37:EC:AE:CF:64:CF:DF:E3:02:2C:A3:7E:CE:B9:05:10:
        3B:FB:6E:19:1D:3E:F0:4A:01:0F:C3:5F:14:9C:06:70:
        C2:DB:D5:72:1F:47:FD:A7:FC:D7:6A:4E:64:FA:A9:19:
        EE:09:24:F8:B9:C4:01:12:D1:3C:AD:89:05:61:94:39:
        67:6D:8D:F1:03:C1:8E:23:6F:A0:EE:6A:B3:C5:FD:C4:
        CC:E6:6C:9A:E3:4B:23:70:18:91:E5:50:10:CB:87:2A:
        A0:54:BC:DD:97:CD:14:C9:CA:05:40:F3:E9:F2:F8:B8:
        FF:ED:BF:7B:5F:85:15:15:23:B4:68:0D:13:56:1C:E4:
        37:6B:4D:E4:93:DE:1E:67:03:FB:92:86:D4:11:1E:3F:
        1F:CD:28:C4:FF:4E:F3:13:23:05:73:F2:00:98:25:77:
        C5:69:22:40:AD:AE:1C:90:6B:60:12:85:61:51:19:ED:
        82:50:EA:C3:BF:AB:F9:54:76:FD:C7:E7:1E:98:A8:45:
        8C:AA:A1:09:87:1A:EE:E8:F4:61:1C:A5:3C:59:E1:02:
        BB:E6:79:3B:EC:1F:21:AE:C1:38:28:F3:F8:05:6F:2C
    FingerPrint
Certificate:
    Data:
      Version:  v3
      Serial Number: 0x1
      Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Issuer: CN=Certificate Authority,OU=pki-ca,O=SjcRedhat Domain 0124
      Validity:
        Not Before: Monday, January 24, 2011 3:56:12 PM PST America/Los_Angeles
        Not  After: Thursday, January 24, 2019 3:56:12 PM PST America/Los_Angeles
      Subject: CN=Certificate Authority,OU=pki-ca,O=SjcRedhat Domain 0124
      Subject Public Key Info:
        Algorithm: RSA - 1.2.840.113549.1.1.1
        Public Key:
          Exponent: 65537
          Public Key Modulus: (2048 bits) :
            D9:51:67:F6:FC:14:59:FF:F0:58:15:5D:78:38:B1:C4:
            AE:CD:38:19:EB:5F:EE:A6:27:B6:CE:DD:27:FE:16:36:
            78:C4:58:87:5D:AD:96:87:17:93:AE:16:60:44:44:BC:
            A0:05:2C:5F:29:60:C7:E8:67:39:5D:CA:5A:7D:BC:45:
            82:84:45:9D:F0:EA:8E:33:AA:5C:29:1A:71:9E:98:4D:
            48:EB:13:94:01:61:F9:06:C3:AE:7B:EE:B0:28:3A:DE:
            27:6B:E1:62:E2:10:FF:76:75:76:44:A4:8B:35:9B:46:
            C2:5E:72:8E:17:65:62:A3:40:97:ED:53:3C:5F:68:87:
```

```
            41:52:8E:A0:84:33:A2:FB:25:29:CF:B1:B7:FE:69:FC:
            7C:BD:BC:50:C5:F8:BC:9F:E5:D0:67:4F:74:6C:72:DF:
            D3:F9:9D:69:36:71:DE:4B:96:E7:BF:6F:28:C5:74:5D:
            4A:79:EB:3C:F1:04:54:3A:B6:8E:3F:A4:D6:18:DE:E1:
            BD:5A:21:5B:84:C8:68:1C:29:6A:E5:9F:26:57:49:E8:
            B8:9D:8D:16:0E:61:2F:FC:B8:28:C7:AA:3E:43:25:44:
            98:B3:32:75:68:FC:1B:97:74:5E:B7:90:65:89:1E:F7:
            E5:D2:E8:E1:4D:6A:1F:1F:15:52:D6:F8:60:9E:AA:D9
    Extensions:
      Identifier: Authority Key Identifier - 2.5.29.35
        Critical: no
        Key Identifier:
          D7:40:65:BA:46:22:D2:7D:63:1E:A0:C8:BF:0B:AD:FC:
          0D:2A:74:7B
      Identifier: Basic Constraints - 2.5.29.19
        Critical: yes
        Is CA: yes
        Path Length Constraint: UNLIMITED
      Identifier: Key Usage: - 2.5.29.15
        Critical: yes
        Key Usage:
          Digital Signature
          Non Repudiation
          Key CertSign
          Crl Sign
      Identifier: Subject Key Identifier - 2.5.29.14
        Critical: no
        Key Identifier:
          D7:40:65:BA:46:22:D2:7D:63:1E:A0:C8:BF:0B:AD:FC:
          0D:2A:74:7B
      Identifier: 1.3.6.1.5.5.7.1.1
        Critical: no
        Value:
          30:34:30:32:06:08:2B:06:01:05:05:07:30:01:86:26:
          68:74:74:70:3A:2F:2F:70:61:77:2E:73:6A:63:2E:72:
          65:64:68:61:74:2E:63:6F:6D:3A:39:31:38:30:2F:63:
          61:2F:6F:63:73:70
  Signature:
    Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
    Signature:
      11:F1:1A:C9:D3:48:CC:43:BA:71:41:27:C3:D4:37:68:
      B7:F9:51:C4:50:FC:AF:B0:82:07:EF:B4:13:77:DE:6A:
      10:0F:A6:7A:32:63:86:89:08:7F:B3:66:E0:C0:3B:58:
      82:80:A7:C2:19:0B:96:94:20:81:96:76:47:F4:38:6F:
      B2:D5:E6:DC:C7:07:5F:AF:E3:42:D9:B0:2E:1B:43:A0:
      4F:91:42:17:E6:0C:E8:09:93:49:88:D9:4C:E0:17:3E:
      B6:0F:76:31:62:18:2A:3E:73:1E:BE:B2:82:18:89:01:
      3A:FC:EE:F0:58:53:17:94:39:31:FB:77:98:BD:5F:A4:
      C8:0A:C7:FC:00:46:F9:D1:63:05:3D:AA:16:F4:67:2D:
      4F:07:A5:55:89:60:83:9E:9D:14:2D:F0:63:CD:5B:74:
      40:6B:D1:22:03:97:59:56:F7:B5:AE:A0:F5:3E:26:1D:
      4E:03:40:00:35:C6:25:3A:1E:63:F5:FC:8E:96:CA:9C:
      75:45:E1:3D:05:2C:4E:AD:8E:BB:6D:23:A4:EA:E1:D0:
      0C:3A:08:7A:50:DD:2E:2B:60:94:A0:EC:C1:1B:23:79:
      50:14:E6:A9:01:F3:AA:AE:5D:5E:DE:47:E0:A2:0A:08:
      20:44:26:84:E3:AB:8E:95:27:41:CF:BD:0B:B9:7F:80
```

```
        FingerPrint


Number of controls is 1
Control #0: CMCStatusInfo
  OID: {1 3 6 1 5 5 7 7 1}
  BodyList: 4164110943
  Status: SUCCESS
```

第15章 CMCREVOKE （失効要求への署名）

CMC Revocation 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します。

15.1. SYNTAX

このユーティリティーの構文は以下のとおりです。

> **CMCRevoke -d***directoryName* **-h***password* **-n***nickname* **-i***issuerName* **-s***serialNumber* **-m***reasonToRevoke* **-c***comment*

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。

注記

引用符で囲まれたスペースを含む値を囲みます。

オプション	説明
c	リクエストに関するテキストコメント。
d	エージェント証明書を含む **cert8.db**、**key3.db**、および **secmod.db** データベースがあるディレクトリーへのパス。これは通常、ホームディレクトリーのブラウザー証明書データベースなどのエージェントの個人ディレクトリーです。
h	エージェントの証明書を含む NSS データベースにアクセスするためのパスワード。
i	失効する証明書の発行者名。
m	証明書が取り消される理由。許可されるさまざまな失効理由の理由コードは以下のとおりです。 ● 0 - 指定されていません。 ● 1: 侵害された鍵 ● 2 - CA キーが侵害されました。 ● 3 - 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 - 操作の停止。 ● 6: 証明書が保留中です。

オプション	説明
n	エージェントの証明書のニックネーム。
s	失効する証明書の 10 進数のシリアル番号。

注記

引用符で囲まれたスペースを含む値を囲みます。

15.2. CMC 失効のテスト

以下を実行して、CMC 失効が適切に機能していることを確認します。

1. 既存の証明書の CMC 失効要求を作成します。たとえば、エージェント証明書を含むディレクトリーが ~jsmith/.mozilla/firefox/ の場合、証明書のニックネームは CertificateManagerAgentCert で、証明書のシリアル番号は 22 の場合、コマンドは以下のようになります。

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

2. CA のエンドエンティティーを開きます。
3. Revocation タブを選択します。
4. メニューで CMC Revoke リンクを選択します。
5. CMCRevoke 操作からテキストボックスに出力を貼り付けます。貼り付けたコンテンツから -----BEGIN NEW CERTIFICATE REQUEST----- 行および -----END NEW CERTIFICATE REQUEST----- 行を削除します。
6. 送信 をクリックします。
7. 結果ページには、証明書 22 が取り消されたことが表示されます。

15.3. 出力

CMCRevoke はエージェントの証明書によって署名された失効要求を生成するため、返される出力は証明書 要求 の形式になります。以下に例を示します。

```
# CMCRevoke -d"~jsmith/.mozilla/firefox" -n"CA Administrator of Instance pki-ca Example
Domain ID" -i"CN=Certificate Authority,OU=pki-ca,O=Example Domain" -s22 -m6 -hsecret -
c"test comment"
cert/key prefix =
path = .
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIJcgYJKoZIhvcNAQcCoIIJYzCCCV8CAQMxCzAJBgUrDgMCGgUAMIHQBggrBgEF
BQcMAqCBwwSBwDCBvTCBtDAtAgEBBggrBgEFBQcHBjEeBBxNZ3l1eFFzc0VWRDJW
SlE4U0VuVFVXYzEvSjg9MIGCAgECBggrBgEFBQcHETFzMHEwUTEeMBwGA1UEChMV
U2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8wDQYDVQQLEwZwa2ktY2ExHjAcBgNVBAMT
FUNlcnRpZmljYXRlIEF1dGhvcml0eQIBFgoBBgQIbmV0c2NhcGUMDHRlc3QgY29t
```

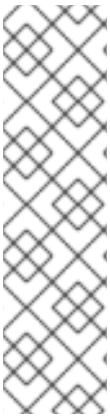
bWVudDAAMAAwAKCCBzgwggNqMIICUqADAgECAgEGMA0GCSqGSIb3DQEBCwUAMFEx
HjAcBgNVBAoTFVNqY1JlZGhhdCBEb21haW4gMDEyNDEPMA0GA1UECxMGcGtpLWNh
MR4wHAYDVQQDExVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkwHhcNMTEwMTI0MjM1NzE1
WhcNMTMwMTEzMjM1NzE1WjCBiTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAw
MTI0MR0wGwYJKoZIhvcNAQkBFg5jZnVAcmVkaGF0LmNvbTEVMBMGCgmSJomT8ixk
AQETBWFkbWluMTEwLwYDVQQDEyhDQSBBZG1pbmlzdHJhdG9ylG9mIEluc3RhbmNl
IHBraS1jYS0wMTI0MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDRrkJOsVDJ
LmPl9fzcVwQRM9Ddc2YM4qhgzC5kKK2MVZJQ7AjMfbWo6t/6/btggy2iNEE721MD
3dVfggz36KRxtXkzi+tm8y3RdfO+EPDltPIA/ZuiBWIJ2NLaKiSS1P9XjiVD5G2J
xicJnxtzE1m6a9jKxK/sMGpT4JsrZWjs8QIDAQABo4GXMIGUMB8GA1UdIwQYMBaA
FNdAZbpGltJ9Yx6gyL8LrfwNKnR7MEIGCCsGAQUFBwEBBDYwNDAyBggrBgEFBQcw
AYYmaHR0cDovL3Bhdy5zamMucmVkaGF0LmNvbTo5MTgwL2NhL29jc3AwDgYDVR0P
AQH/BAQDAgTwMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDBDANBgkqhkiG
9w0BAQsFAAOCAQEAsEBJo41ZpoBHQDvxGGk0F5n0YRLDGLg4rpuhKCgqQkG+CUXl
F3pMayZ06U8JFGyausYrg6M6yx68lgJ5NWOlh8SC1JCaaQrwol3FZlnL1XhlNkuq
flSj2wuSy04aLTdTiFO+DfH3RPs7pxVfsZz/+BpCUCrjSYkU2ThZr4f42ECKgwXD
T4VX3LttNB9VGh6i8RttoBAcsarP5mtdzBELWjGQQL4nHOlub4U9NWME9rLmHCQ5
mKWlISk417Zue9pTYDGDKTY8Z0aMlnKuRM/Kl+QmXN/Xwd5d7vHwQkc6qD5Q1UGR
czT/scr88iUYVHOaHoCGnmpTnYdvF/Tl3vnhPDCCA8YwggKuoAMCAQICAQEwDQYJ
KoZIhvcNAQELBQAwUTEeMBwGA1UEChMVU2pjUmVkaGF0IERvbWFpbiAwMTI0MQ8w
DQYDVQQLEwZwa2ktY2ExHjAcBgNVBAMTFUNlcnRpZmljYXRlIEF1dGhvcml0eTAe
Fw0xMTAxMjQyMzU2MTJaFw0xOTAxMjQyMzU2MTJaMFExHjAcBgNVBAoTFVNqY1Jl
ZGhhdCBEb21haW4gMDEyNDEPMA0GA1UECxMGcGtpLWNhMR4wHAYDVQQDExVDZXJ0
aWZpY2F0ZSBBdXRob3JpdHkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQDZUWf2/BRZ//BYFV14OLHErs04Getf7qYnts7dJ/4WNnjEWIddrZaHF5OuFmBE
RLygBSxfKWDH6Gc5XcpafbxFgoRFnfDqjjOqXCkacZ6YTUjrE5QBYfkGw6577rAo
Ot4na+Fi4hD/dnV2RKSLNZtGwl5yjhdIYqNAI+1TPF9oh0FSjqCEM6L7JSnPsbf+
afx8vbxQxfi8n+XQZ090bHLf0/mdaTZx3kuW579vKMV0XUp56zzxBFQ6to4/pNYY
3uG9WiFbhMhoHClq5Z8mV0nouJ2NFg5hL/y4KMeqPkMlRJizMnVo/BuXdF63kGWJ
HvfI0ujhTWofHxVS1vhgnqrZAgMBAAGjgagwgaUwHwYDVR0jBBgwFoAU10BlukYi
0n1jHqDlvwut/A0qdHswDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAcYw
HQYDVR0OBBYEFNdAZbpGltJ9Yx6gyL8LrfwNKnR7MEIGCCsGAQUFBwEBBDYwNDAy
BggrBgEFBQcwAYYmaHR0cDovL3Bhdy5zamMucmVkaGF0LmNvbTo5MTgwL2NhL29j
c3AwDQYJKoZIhvcNAQELBQADggEBABHxGsnTSMxDunFBJ8PUN2i3+VHEUPyvsIlH
77QTd95qEA+mejJjhoklf7Nm4MA7WIKAp8lZC5aUlIGWdkf0OG+y1ebcxwdfr+NC
2bAuG0OgT5FCF+YM6AmTSYjZTOAXPrYPdjFiGCo+cx6+solYiQE6/O7wWFMXIDkx
+3eYvV+kyArH/ABG+dFjBT2qFvRnLU8HpVWJYlOenRQt8GPNW3RAa9EiA5dZVve1
rqD1PiYdTgNAADXGJToeY/X8jpbKnHVF4T0FLE6tjrttl6Tq4dAMOgh6UN0uK2CU
oOzBGyN5UBTmqQHzqq5dXt5H4KIKCCBEJoTjq46VJ0HPvQu5f4AxggE8MIIBOAIB
AzBWMFExHjAcBgNVBAoTFVNqY1JlZGhhdCBEb21haW4gMDEyNDEPMA0GA1UECxMG
cGtpLWNhMR4wHAYDVQQDExVDZXJ0aWZpY2F0ZSBBdXRob3JpdHkCAQYwCQYFKw4D
AhoFAKA+MBcGCSqGSIb3DQEJAzEKBggrBgEFBQcMAjAjBgkqhkiG9w0BCQQxFgQU
OuuE/YrvX8NcpqmrylTJBibBpAAwDQYJKoZIhvcNAQEBBQAEgYBfBvFP7b/LIUZd
FJpqPguPaSgrulo7wLR0rnFki/HacTMg9Eh+b1KnHopeCYTvOlrC10li1F8EEw9J
yBMt+JHfM6P6hOulo3PF9ciYdEmPBdLHv2itMnU3jeTHfU8qmV1rKkt1VXL47M1Y
HPWxykQ+R28Eet/qPC6dNVqc8AWuqQ==
-----END NEW CERTIFICATE REQUEST-----

第16章 CRMFPOPCLIENT (ENCODED CRMF 要求の送信)

CRMFPopClient ユーティリティーは、CA サーバーで検証できる POP (POP)データでエンコードされた要求を使用して、Certificate Request Message Format (CRMF)要求を Certificate System CA に送信するツールです。クライアントが要求で POP 情報を提供する場合、サーバーは要求元が新しい証明書の秘密鍵を所有していることを確認できます。

このツールは、次のすべてを実行します。

1. **CA は CRMF 要求内にエンコードされた POP 情報を強制または検証します。**
2. **標準の Certificate System エージェントページまたはインターフェイスを使用せずに、簡単な証明書要求を行います。**
3. **KRA からのキーアーカイブのトランスポート証明書を含む簡単な証明書要求を作成します。**



注記

KRA のトランスポート証明書を含む transport.txt ファイルは、コマンドを実行するディレクトリーに存在する必要があります。ファイルがない場合、アーカイブプロセスは試行されますが、以下のエラーメッセージで失敗します。

```
ERROR: File 'transport.txt' does not exist
Try 'CRMFPopClient --help' for more information.
```

transport.txt には、ヘッダーとフッターが削除された 1 行で、ベース 64 でエンコードされたトランスポート証明書全体が必要です。

16.1. SYNTAX

用途に応じて、CRMFPopClient ユーティリティーには 2 つの構文スタイルがあります。

これは、簡単な証明書要求を CA に送信します。

CRMFPopClient *token_password profile_name ホスト port username requester_name pop_option subject_dn* [OUTPUT_CERT_REQ]

これは、証明書要求を CA に送信せずに標準出力(stdout)に出力するためのものです。

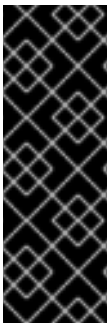
CRMFPopClient *token_password pop_option* OUTPUT_CERT_REQ *subject_dn*

オプション	説明
token_password	暗号化トークンのパスワード。
profile_name	要求を送信する CA プロファイル。
ホスト	CA インスタンスのホスト名。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。

オプション	説明
port	Certificate System CA の SSL ポート以外のポート。
username	証明書要求の発行先の Certificate System ユーザー。
requester_name	証明書を要求する人またはエンティティーの名前。
pop_option	生成する POP 要求のタイプを設定します。無効な要求を生成できるため、このオプションはテストに使用できます。以下の 3 つの値があります。 • **POP_SUCCESS**.正しい POP 情報で要求を生成します。サーバーは、情報が正しいことを検証します。 • **POP_FAIL**.誤った POP 情報で要求を生成します。サーバーが送信されている場合はこの要求を拒否します。これは、サーバー設定をテストするために使用されます。 • **POP_NONE**POP 情報のない CRMF 要求を生成します。サーバーがすべての POP 情報を検証するように設定されている場合は、この要求を拒否します。その場合は、サーバー設定のテストに使用できます。
subject_dn	要求された証明書の識別名。
OUTPUT_CERT_REQ	生成された証明書要求を画面に出力します。これは、CRMF POP リクエストが CA に送信される場合はオプションですが、コマンドを使用して要求を返す場合は必要になります。

16.2. 使用方法

CRMFPopClient には、生成するリクエストを処理する方法は 2 つあります。CA に直接送信することも、単に要求を stdout に出力することもできます。



重要

base-64 形式のトランスポート証明書を含む **transport.txt** という名前のファイルを、ユーティリティーの起動元のディレクトリーに作成 する必要 があります。このファイルは、KRA へのアーカイブに利用できる必要があります。ファイルが存在する場合、ツールはこのファイルを自動的に選択し、キーのアーカイブを実行します。

transport.txt には、ヘッダーとフッターが削除された 1 行で、ベース 64 でエンコードされたトランスポート証明書全体が必要です。

CA への要求の送信

以下の例では、CRMF/POP 要求を生成し、サーバーが正しいことを確認し、証明書要求を画面に出力します。

```
CRMFPopClient secret caUserCert host.example.com 1026 CaUser jsmith POP_SUCCESS
CN=MyTest,C=US,UID=CaUser
```

要求を標準出力(stdout)に出力する

リクエストは単に stdout に出力できます。これは、証明書要求が外部 CA に送信される場合や、CA の Web サービスページから直接送信される場合に便利です。リクエストを処理するには、追加の手動手順が必要です。

1. CRMFPopClient を使用して、POP データでエンコードされた要求を生成します。

```
CRMFPopClient secret POP_SUCCESS OUTPUT_CERT_REQ
CN=MyTest,C=US,UID=CaUser
```

2. 画面に出力されるリクエストをコピーします。

```
MIIFczCCBW8wggTVAgEBMIHygAECpUswSTEaMBgGCgmSJomT8ixkAQETCmptYWdu
ZUNSTUYxCzAJBgNVBAYTAlVTMR4wHAYDVQQDExVqbWFnbmVDUk1GYXJjaGl2ZVRl
c3SmgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJiLbrQaChfzBQLnEnehA3uj
01dA0+pBIJH5PHngjeRpXc6XyYnRpQuFriZUKW7QXewUYQbYsB13F8OwGADfS8wZ
zxfBvLqvQb7h9JtLdsHMVXbQ69/cEs/jCU5Cmr1LmFs4EAAO9Yr/CJjp2hscY82e
KdyGEB6pWuXuBprc8lRJAgMBAAEwggPZMIIDswYJKwYBBQUHBQEEooIDpDCCA6Ch
FAYIKoZIhvcNAwcECAEBAQEBAQEBgoIBAQBwc6w+H6qZKqQSzQZAOIBc97UowcjfYH/vqGsSiN7bkFzx9kEWBZ6hlxP8gY/2JxJQsD0lxsykXcdlC6pW3GwGnBI7obM7
eKeNwL0Mi22ANXdkP7l6KFPFIMNg5v0bynCKOYr2n+ZRQEXnGdLHWnG+vh2GGpDH
1ocXV46dFqeCnSpVEXS/PCcS4l65hByRFMU8lB5vPPBnNJxJt4jY6FU209Y+mrEd
8J2dmtqYLo7y4BhzbBfPn08O1QFJXWGi6ZUblirZlnv4Fg+us1gdlM1wVJSr4rNu
oZx6+JT40ZJ7i0k63T/jMvW77oQesFG21MCOvxrYZJTgTXZ9+sqlKZ/zA4lCgQB6
Dm/JGjAOKdPdpKW1zYs6hpJsjQsLTM5Mz1ONFn7DLe9RDuXdpWOpyjBcqyNqC47Y
CQkRPMW4kj/7XgR4lmycEZZD8OtJF3MqTP7JQGmEXHdsiLRRQy0w/tm0lyl7rJ5p
F34hualY0xtbO+GfaKuUB2GH59Zy11oRug1Okm1UQb/HYuCTL0gh6wH4TXk/g6sx
WVv4cegqsdaZpqAG9+BqvLw9t5R+8dsCCpUTVRg7lIEL9HxSAUF2lon9QEEvQJAD
lvofSSXBBf2w+/Qp1x60ZJl7+0vb9P3gEyR3c+BlblkkdAbfM5knGe2LTnCPcrDb
dY1OV8sgFGxGxcqW2+edJd/yRmsWp/6Dh3HHkd234bUvu+6r5GY7ebueOQlr1HsN
Zwc9XSGLmaShrBTgLyHwq2G3qx7riCCZz6KpSui8YDuQQZE93BoNcuBzvgl/4rlb
uBJfqGYb2t8mSb8Ss+jumbHbZByaVPYp4D9l0Jg3UVbccb19QRlz3G75QotKmDqY
YT7UVbVduLddWN8YvXtoEYcOEfesrdnkEqiHmsALWM0/4U0vWk1Uw7t59O6QMomJ
l8lPc0lZzl1cYaAuuF5SJv/bb/+9S1GqltuuIt5+bi5t5vN4OE02BfHrpZQHkCbn
ezslwhDnlTwYZSxjMzAeZkBzghTRcNrPwXnvx3crNW2tyZo68FoqOlXAYf/uNBdY
lEBdsvgNPzlRwR63u7pqWA9sJc15X/lwPZ8xj49UwB/cCoSt8PGFADPaAWkSMaT2
rv5+LRkcR56Ol3aMjE9OQEN3kRH75oEGyL5jMkkMa58QGtQgs9WnIhwin0TgWYA2
99wD38RcHVogyQ6Nl4y/MCAGCCsGAQUFBwcXBBTmaclfLv+kkK5z5kTMP54dlnec
UKGBkzANBgkqhkiG9w0BAQQFAAOBgQAqY9mrSqcjPSP9M8p8/TVWdlXn982styAT
DEdau50jksjO/LHPheeFUlaf4+SamE5SUMcEJH9R2p9dqZN8JpvgCYn+h8rjKnlM
5mKstkjtOj42mwizvphkaxlMZdrTSbfC0QjCmkjP2yl3F5QbOoowZ9REH4BMLqRU
sLTu2xgVrw==
```

3. CA のエンドエンティティーを開きます。

```
https://server.example.com:9444/ca/ee/ca/
```

4. CRMF/POP 要求をサポートするすべてのプロファイルに要求を送信します。

16.3. 出力

CRMFPopClient コマンドが正常に実行されると、CA は証明書要求のベース 64 でエンコードされたバージョンを送信し、返される HTML ページには CA Web フォームを介して要求が送信されました。

```
CRMFPopClient password caUserCert test.example.com 9180  joeCRMF joeCRMF
POP_SUCCESS "CN=joeCRMFarchiveTest,C=US, UID=joeCRMF"  OUTPUT_CERT_REQ

Proof Of Possession Utility....

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
Generating Legal POP Data.....
.
Signature completed...

Generated Cert Request: ......

MIIFczCCBW8wggTVAgEBMIHygAECpUswSTEaMBgGCgmSJomT8ixkAQETCmptYWdu
ZUNSTUYxCzAJBgNVBAYTAlVTMR4wHAYDVQQDExVqbWFnbmVDUk1GYXJjaGl2ZVRl
c3SmgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJiLbrQaChfzBQLnEnehA3uj
01dA0+pBlJH5PHngjeRpXc6XyYnRpQuFriZUKW7QXewUYQbYsB13F8OwGADfS8wZ
zxfBvLqvQb7h9JtLdsHMVXbQ69/cEs/jCU5Cmr1LmFs4EAAO9Yr/CJjp2hscY82e
KdyGEB6pWuXuBprc8IRJAgMBAAEwggPZMIIDswYJKwYBBQUHBQEEooIDpDCCA6Ch
FAYIKoZIhvcNAwcECAEBAQEBAQEBgoIBAQBwc6w+H6qZKqQSzQZAOlBc97Uowcjf
YH/vqGsSiN7bkFzx9kEWBZ6hlxP8gY/2JxJQsD0lxsykXcdlC6pW3GwGnBl7obM7
eKeNwL0Mi22ANXdkP7l6KFPFlMNg5v0bynCKOYr2n+ZRQEXnGdLHWnG+vh2GGpDH
1ocXV46dFqeCnSpVEXS/PCcS4l65hByRFMU8lB5vPPBnNJxJt4jY6FU209Y+mrEd
8J2dmtqYLo7y4BhzbBfPn08O1QFJXWGi6ZUblirZlnv4Fg+us1gdlM1wVJSr4rNu
oZx6+JT40ZJ7i0k63T/jMvW77oQesFG21MCOvxrYZJTgTXZ9+sqlKZ/zA4lCgQB6
Dm/JGjAOKdPdpKW1zYs6hpJsjQsLTM5Mz1ONFn7DLe9RDuXdpWOpyjBcqyNqC47Y
CQkRPMW4kj/7XgR4lmycEZZD8OtJF3MqTP7JQGmEXHdsiLRRQy0w/tm0lyl7rJ5p
F34hualY0xtbO+GfaKuUB2GH59Zy11oRug1Okm1UQb/HYuCTL0gh6wH4TXk/g6sx
WVv4cegqsdaZpqAG9+BqvLw9t5R+8dsCCpUTVRg7lIEL9HxSAUF2lon9QEEvQJAD
IvofSSXBBf2w+/Qp1x60ZJl7+0vb9P3gEyR3c+BlblkkdAbfM5knGe2LTnCPcrDb
dY1OV8sgFGxGxcqW2+edJd/yRmsWp/6Dh3HHkd234bUvu+6r5GY7ebueOQlr1HsN
Zwc9XSGLmaShrBTgLyHwq2G3qx7riCCZz6KpSui8YDuQQZE93BoNcuBzvgl/4rlb
uBJfqGYb2t8mSb8Ss+jumbHbZByaVPYp4D9l0Jg3UVbccb19QRlz3G75QotKmDqY
YT7UVbVduLddWN8YvXtoEYcOEfesrdnkEqiHmsALWM0/4U0vWk1Uw7t59O6QMomJ
l8lPc0lZzl1cYaAuuF5SJv/bb/+9S1GqltuuIt5+bi5t5vN4OE02BfHrpZQHkCbn
ezslwhDnlTwYZSxjMzAeZkBzghTRcNrPwXnvx3crNW2tyZo68FoqOlXAYf/uNBdY
```

```
lEBdsvgNPzlRwR63u7pqWA9sJc15X/lwPZ8xj49UwB/cCoSt8PGFADPaAWkSMaT2
rv5+LRkcR56Ol3aMjE9OQEN3kRH75oEGyL5jMkkMa58QGtQgs9WnIhwin0TgWYA2
99wD38RcHVogyQ6NI4y/MCAGCCsGAQUFBwcXBBTmaclfLv+kkK5z5kTMP54dInec
UKGBkzANBgkqhkiG9w0BAQQFAAOBgQAqY9mrSqcjPSP9M8p8/TVWdlXn982styAT
DEdau50jksjO/LHPheeFUlaf4+SamE5SUMcEJH9R2p9dqZN8JpvgCYn+h8rjKnIM
5mKstkjtOj42mwizvphkaxlMZdrTSbfC0QjCmkjP2yl3F5QbOoowZ9REH4BMLqRU
sLTu2xgVrw==

End Request:

Server Response.....
--------------------

<!-- --- BEGIN COPYRIGHT BLOCK ---
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; version 2 of the License.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

    Copyright (C) 2007 Red Hat, Inc.
    All rights reserved.
    --- END COPYRIGHT BLOCK --- -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<script type="text/javascript">
errorReason="Request Deferred - defer request";
requestListSet = new Array;
requestList = new Object;
requestList.requestId="284";
requestListSet[0] = requestList;
errorCode="2";
</script>
<font size="+1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-serif">
Certificate Profile
</font><br>
  <Font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-serif">
<p>
</font>
<table border="0" cellspacing="0" cellpadding="0"
background="/ca/ee/graphics/hr.gif"
width="100%">
  <tr>
    <td> </td>
  </tr>
</table>
<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-serif">
<script language=javascript>
```

```
var autoImport = 'false';

if (errorCode == 0) { // processed
  document.write('Congratulations, your request has been processed successfully
');
  document.writeln('<P>');
  for (var i = 0; i < requestListSet.length; i++) {
    document.write('Your request ID is ');
    document.write('<B>'+requestListSet[i].requestId+'</B>.');
    document.writeln('<P>');
  }
  document.writeln('<b>');
  document.writeln('Outputs');
  document.writeln('</b>');
  document.writeln('<P>');
  document.writeln('<table width=100%>');
for (var i = 0; i < outputListSet.length; i++) {
    document.writeln('<tr valign=top>');
    document.writeln('<td>');
    document.writeln('<FONT size="-1" face="PrimaSans BT, Verdana,
sans-serif">'
);
    document.writeln('<li>');
    document.writeln(outputListSet[i].outputName);
    document.writeln('</FONT>');
    document.writeln('</td>');
    document.writeln('<tr valign=top>');
    document.writeln('</tr>');
    document.writeln('<td>');
    if (outputListSet[i].outputSyntax == 'string') {
      document.writeln(outputListSet[i].outputVal);
    } else if (outputListSet[i].outputSyntax == 'pretty_print') {
      document.writeln('<pre>');
      document.writeln(outputListSet[i].outputVal);
      document.writeln('</pre>');
    }
    document.writeln('</td>');
    document.writeln('</tr>');
}
   document.writeln('</table>');
   document.writeln('<p>');
  document.writeln('<table width=100%>');
    document.writeln('<tr valign=top>');
    document.writeln('<td>');
    document.writeln('<FONT size="-1" face="PrimaSans BT, Verdana,
sans-serif">'
);
    document.writeln('<li>');
    document.writeln('Certificate Imports');
    document.writeln('</FONT>');
    document.writeln('</td>');
   for (var i = 0; i < requestListSet.length; i++) {
    document.writeln('<tr valign=top>');
    document.writeln('<td>');
if (autoImport == 'true') {
```

```
    // only support one certificate import
   var loc = "getCertFromRequest?requestId="+ requestListSet[i].requestId +
"&importCert=true";
   document.write("<iframe width='0' height='0' src='"+loc+"' </iframe>");
} else {
    document.writeln('<form method=post action="getCertFromRequest">');
 if (navigator.appName == "Netscape") {
    document.writeln('<input type=hidden name=importCert value=true>');
 } else {
    document.writeln('<input type=hidden name=importCert value=false>');
 }
    document.writeln('<input type=hidden name=requestId value=' +
requestListSet[i].requestId + '>');
    document.writeln('<input type=submit name="Import Certificate"
value="Import Certificate">');
    document.writeln('</form>');
}
    document.writeln('</td>');
    document.writeln('</tr>');
   }
   document.writeln('</table>');
} else if (errorCode == 1) { // not submitted
  document.write('Sorry, your request is not submitted. The reason is "' +
errorReason + '".');
} else if (errorCode == 2) { // pending
  document.write('Congratulations, your request has been successfully ');
  document.write('submitted. ');
  document.write('Your request will be processed when an authorized agent ');
  document.writeln('verifies and validates the information in your request.');
  document.writeln('<P>');
  for (var i = 0; i < requestListSet.length; i++) {
    document.write('Your request ID is ');
    document.write('<B><a href="checkRequest?requestId=');
    document.write(requestListSet[i].requestId);
    document.write('">'+requestListSet[i].requestId+'</a></B>.');
    document.writeln('<P>');
  }
  document.write('Your can check on the status of your request with ');
  document.write('an authorized agent or local administrator ');
  document.writeln('by referring to this request ID.');
} else if (errorCode == 3) { // rejected
  document.write('Sorry, your request has been rejected. The reason is "' +
errorReason + "");
  document.writeln('<P>');
  for (var i = 0; i < requestListSet.length; i++) {
    document.write('Your request ID is ');
    document.write('<B>'+requestListSet[i].requestId+'</B>.');
    document.writeln('<P>');
  }
} else { // unknown state
  document.write('Sorry, your request is not submitted. The error code is "' +
errorReason + '".');
}
</script>
</font>
</html>
```

第17章 EXTJOINER（リクエストへの拡張機能の追加）

Certificate System は、サーバーが発行するエンドエンティティー証明書に、標準およびカスタム X.509 証明書拡張を追加できるようにするポリシープラグインモジュールを提供します。同様に、サブシステムユーザーの証明書を生成する Certificate Setup Wizard を使用すると、拡張機能を選択し、証明書に追加できます。エージェントインターフェイスのウィザードインターフェイスと request-approval ページには、その MIME-64 でエンコードされた形式でエクステンションを貼り付けるテキスト領域が含まれます。

エクステンションを貼り付けるためのテキストフィールドは、単一の拡張 BLOB を受け入れます。複数の拡張機能を追加するには、まず単一の拡張ブロブにまとめて、テキストフィールドに貼り付ける必要があります。**ExtJoiner** ツールは、複数の拡張機能を単一の MIME-64 でエンコードされたブロブに結合します。この新しい組み合わせた BLOB をウィザードテキストフィールドに貼り付けるか、エージェントインターフェイスの request-approval ページに貼り付けて、一度に複数の拡張機能を指定できます。

17.1. SYNTAX

ExtJoiner ユーティリティーの構文は以下のようになります。

> **ExtJoiner** *ext_file0 ext_file1 ... ext_fileN*

オプション	説明
ext_file#	X.509 拡張の base-64 DER エンコーディングを含むファイルのパスと名前を指定します。

17.2. 使用方法

ExtJoiner は MIME-64 でエンコードされた形式で拡張を生成 しません。既存の MIME-64 でエンコードされた拡張機能に参加します。複数のカスタム拡張機能に参加し、**ExtJoiner** を使用して証明書要求に拡張機能を追加するには、以下の手順を実施します。

1. 拡張プログラムファイルの場所を見つけ、書き留めておきます。
2. 拡張ファイルを指定して、**ExtJoiner** を実行します。たとえば、**/etc/extensions** というディレクトリーに **myExt1** および **myExt2** という名前の拡張子ファイルがある場合、コマンドは以下のようになります。

   > **ExtJoiner /etc/extensions/myExt1 /etc/extensions/myExt2**

   これにより、以下の例のように、結合された拡張機能の base-64 でエンコードされた Blob が作成されます。

   > **MEwwLgYDVR0IAQHBCQwIgYFKoNFBAMGCIGC5EKDM5PeXzUGBi2CVyLNCQYFU**
   > **iBakowGgYDVR0SBBMwEaQPMA0xCzAJBgNVBAYTAIVT**

3. 変更せずにエンコードされた BLOB をファイルにコピーします。
4. **AtoB** ユーティリティーを使用してバイナリーデータを ASCII に変換し、**dumpasn1** ユーティリティーを使用して base-64 でエンコードされたブロブの内容をダンプして、拡張機能が証明書要求に追加する前に適切に結合されていることを確認します。**AtoB** ユーティリティーの詳細

は、*7章AtoB (ASCII からバイナリーへの変換)* を参照してください。**dumpasn1** ツールは、http://fedoraproject.org/extras/4/i386/repodata/repoview/dumpasn1-0-20050404-1.fc4.html からダウンロードできます。

a. **AtoB** ユーティリティーを実行して、ASCII をバイナリーに変換します。

```
AtoB input_file output_file
```

input_file は ASCII の base-64 でエンコードされたデータを含むパスおよびファイルに、output_file はバイナリー出力を書き込むユーティリティーのパスおよびファイルです。

b. **dumpasn1** ユーティリティーを実行します。

```
dumpasn1output_file
```

output_file は、バイナリーデータを含むパスおよびファイルに置き換えます。出力は以下のようになります。

```
0 30 76: SEQUENCE {
2 30 46: SEQUENCE {
4 06 3: OBJECT IDENTIFIER extKeyUsage (2 5 29 37)
9 01 1: BOOLEAN TRUE
12 04 36: OCTET STRING
 : 30 22 06 05 2A 83 45 04 03 06 0A 51 82 E4 42 83
 : 33 93 DE 5F 35 06 06 2D 82 57 22 CD 09 06 05 51
 : 38 81 6A 4A
 : }
50 30 26: SEQUENCE {
52 06 3: OBJECT IDENTIFIER issuerAltName (2 5 29 18)
57 04 19: OCTET STRING
 : 30 11 A4 0F 30 0D 31 0B 30 09 06 03 55 04 06 13
 : 02 55 53
 : }
 : }

0 warnings, 0 errors.
```

出力データが正しく表示されないように見える場合は、元の Java™ 拡張ファイルが正しいことを確認し、ASCII からバイナリーへの変換と、正しい出力が返されるまでデータのダンプを繰り返します。

5. 拡張機能が検証されたら、**ExtJoiner** を実行して作成した base-64 でエンコードされた BLOB を Certificate System ウィザード画面にコピーし、証明書または証明書署名要求(CSR)を生成します。

第18章 GENEXTKEYUSAGE （リクエストへのキー使用拡張の追加）

GenExtKeyUsage ツールは、証明書に ExtendedKeyUsage (OID 2.5.29.37)を追加するベース 64 でエンコードされたブロブを作成します。この BLOB は、証明書の作成時に証明書承認ページに貼り付けられます。

18.1. SYNTAX

GenExtKeyUsage ツールの構文は以下のとおりです。

```
GenExtKeyUsage [true|false] OID ...
```

オプション	説明
true | false	重大度を設定します。**true** は、拡張機能が重要であることを意味します。**false** は重要ではないことを意味します。重大度の値は、証明書の検証プロセス中に使用されます。拡張機能がクリティカルとしてマークされている場合は、パス検証ソフトウェアはその拡張機能を解釈できる必要があります。
OID	証明書に選択された各証明書タイプを表す OID 番号。

第19章 GENISSUERALTNAMEEXT （リクエストへの発行者名エクステントの追加）

GenIssuerAltNameExt は、発行者名拡張子 IssuerAltNameExt (OID 2.5.29.18)を新しい証明書に追加するベース 64 でエンコードされたブロブを作成します。この BLOB は、証明書の作成時に証明書承認ページに貼り付けられます。

19.1. SYNTAX

GenIssuerAltNameExt ツールは、パラメーターのペアを使用します。最初のパラメーターは発行者に使用される name 属性の一般的な型を指定し、2 番目のパラメーターはその形式でその名前を指定します。ツールの構文は以下のとおりです。

GenIssuerAltNameExt *general_type# ... general_name# ...*

パラメーター	説明
general_type	名前のタイプを設定します。以下の文字列のいずれかになります。 • RFC822Name • DNSName • EDIPartyName • URIName • IPAddressName • OIDName • X500Name

パラメーター	説明
general_name	発行者の名前を示す name タイプに準拠する文字列。 • **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。 • **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。 • **IPAddressName** の場合、値は有効な IP アドレスである必要があります。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。 • **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。 • **X500Name** の場合、値は証明書のサブジェクト名と同様に 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

19.2. 使用方法

以下の例では、**RFC822Name** および **X500Name** 形式で発行者名を設定します。

```
GenIssuerAltNameExt RFC822Name TomTom@example.com X500Name cn=TomTom
```

第20章 SUBJECTALTNAMEEXT （リクエストへのサブジェクト代替名拡張の追加）

GenSubjectAltNameExt は、base-64 でエンコードされたブロブを作成して、別のサブジェクト名エクステンション SubjectAltNameExt (OID 2.5.29.17)を新しい証明書に追加します。この BLOB は、証明書の作成時に証明書承認ページに貼り付けられます。

20.1. SYNTAX

GenSubjectAltNameExt ツールは、パラメーターのペアを使用します。ここで、最初のパラメーターは名前形式のタイプを指定し、2 番目のパラメーターは指定された形式でその名前を指定します。

このツールの構文は以下のとおりです。

GenSubjectAltNameExt *general_type# ... general_name# ...*

パラメーター	説明
general_type	**使用される名前のタイプを設定します。これには、以下の文字列のいずれかを使用できます。** • **RFC822Name** • **DNSName** • **EDIPartyName** • **URIName** • **IPAddressName** • **OIDName** • **X500Name**

パラメーター	説明
general_name	サブジェクト名の指定された形式に準拠する文字列。 ● **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** ● **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。たとえば、**testCA.example.com** です。 ● **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** ● **IPAddressName** の場合、値は有効な IP アドレスである必要があります。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**、**0:0:0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF: FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** です。 ● **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。 ● **X500Name** の場合、値は証明書のサブジェクト名と同様に X.500 名の文字列形式である必要があります。たとえば、**cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** などです。

20.2. 使用方法

以下の例では、サブジェクトの別名は **RFC822Name** および **X500Name** タイプに設定されます。

```
GenSubjectAltNameExt RFC822Name TomTom@example.com X500Name cn=TomTom
```

第21章 HTTPCLIENT (HTTP でのリクエストの送信)

HTTP Client ユーティリティーである HttpClient は、CMC 要求(CMC Request ユーティリティーを使用して作成された)または PKCS #10 要求を CA に送信します。

21.1. SYNTAX

このユーティリティーは、1つの .cfg 設定ファイルをパラメーターとして取ります。構文は次のとおりです。

HttpClient*/path/to/file.cfg*

.cfg ファイルには以下のパラメーターが含まれます。

パラメーター	説明
ホスト	Certificate System サーバーのホスト名。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。たとえば、**host=server.example.com** です。
port	Certificate System サーバーのポート番号。たとえば、**port=9443** です。
secure	HTTPS 接続の場合は **true**、HTTP 接続の場合は **false** です。たとえば、**secure=true** です。
入力 (input)	登録要求の完全パスおよびファイル名。バイナリー形式である必要があります。たとえば、**input=cmcReqCRMFBin** です。
出力 (output)	バイナリー形式の応答の完全パスおよびファイル名。たとえば、**output=cmcResp** です。
dbdir	**cert8.db、key3.db、および secmod.db** データベースがあるディレクトリーへの完全パス。**secure=false** の場合、このパラメーターは無視されます。たとえば、**dbdir=/usr/bin** です。
clientmode	クライアント認証の場合は **true**、クライアント認証がない場合は **false** です。**secure=false** の場合、このパラメーターは無視されます。たとえば、**clientmode=true** です。
password	**cert8.db** データベースのパスワード。**secure=false** および **clientauth=false** の場合、このパラメーターは無視されます。たとえば、**password=secret** です。

パラメーター	説明
ニックネーム	クライアント証明書のニックネーム。**clientmode=false** の場合、このパラメーターは無視されます。たとえば、**nickname=CS Agent-102504a の 102504a ID** です。
サーブレット (servlet)	完全な CMC 要求を処理するサーブレットの URI。デフォルト値は **/ca/profileSubmitCMCFull** です。たとえば、**servlet=/ca/profileSubmitCMCFull** です。

第22章 OCSPCLIENT (OCSP リクエストの送信)

OCSP 要求ユーティリティー OCSPClient は、RFC 2560 に準拠する OCSP 要求を作成し、それを OCSP サーバーに送信し、OCSP 応答をファイルに保存します。

22.1. SYNTAX

OCSPClient ツールの構文は次のとおりです。

OCSPClient ***ホスト port dbdir ニックネーム serial_number*** **or** ***filename 出力 (output) 時間***

オプション	説明
ホスト	OCSP サーバーのホスト名を指定します。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。
port	OCSP サーバーのエンドユーザーポート番号を指定します。
dbdir	チェックする証明書に署名した CA 証明書が含まれるセキュリティーデータベース(**cert8.db、key3.** db、および **secmod.db**)の場所を指定します。
ニックネーム	CA 証明書のニックネームを指定します。
serial_number または **filename**	シリアル番号を指定するか、またはステータスを確認する証明書の要求が含まれるファイルの名前を指定します。
出力 (output)	DER でエンコードされた OCSP 応答を出力するパスおよびファイルを指定します。
時間	要求を送信する回数を指定します。

第23章 PKCS10CLIENT (PKCS #10 証明書要求の生成)

PKCS #10 ユーティリティー **PKCS10Client** は、セキュリティーデータベースに 1024 ビットの RSA キーペアを生成し、公開鍵で PKCS#10 証明書要求を構築し、要求をファイルに出力します。

PKCS #10 は、RSA が定義する認定要求構文です。CA は、複数のタイプの証明書要求をサポートする場合があります。Certificate System CA は、KEYGEN、PKCS#10、CRMF、および CMC をサポートします。

CA から証明書を取得するには、証明書要求を CA エージェントに送信して承認する必要があります。承認されると、要求用に証明書が作成され、拡張機能などの証明書属性が証明書プロファイルに従って入力されます。

23.1. SYNTAX

PKCS10Client ツールの構文は次のとおりです。

PKCS10Client -p *certDBPassword* **-d** *certDBDirectory* **-o** *outputFile* **-s** *subjectDN*

オプション	説明
p	セキュリティーデータベースのパスワードを指定します。
d	セキュリティーデータベースへのパスを指定します。
o	新しい PKCS #10 証明書をベース 64 形式で出力するパスとファイル名を設定します。
s	証明書のサブジェクト DN を指定します。

第24章 PKCS12EXPORT （データベースからの証明書および鍵をエクスポート）

PKCS12Export は、NSS セキュリティーデータベース内のすべての証明書および対応する鍵を指定された .p12 出力ファイルにダンプします。

24.1. SYNTAX

PKCS12Export -d */path/to/cert-directory* -p *keydb-password-file* -w *pkcs12-password-file* -o *output-file.p12* [-debug]

パラメーター	説明
-d	エクスポートする証明書が含まれる NSS データベースへの完全パスを指定します。
-o	エクスポートされた証明書を出力するファイルの名前を指定します。
-p	NSS セキュリティーデータベースにアクセスするためのパスワードを含むファイルの完全パスおよびファイル名を指定します。
-w	出力ファイルにアクセスするためのパスワードを設定するために使用するファイルの完全パスおよびファイル名を指定します。
-debug	stdout へのデバッグ出力を有効にします。

24.2. 使用方法および出力

PKCS12Export コマンドは、データベースの各証明書を .p12 出力ファイルにエクスポートします。-debug オプションを使用すると、操作を進める際に、各証明書の証明書のニックネームが stdout に出力されます。（そうしないと、コマンドの出力はありません。）

```
# PKCS12Export -debug -d /var/lib/pki-ca/alias -p dbpwd.txt -w p12pwd.txt -o master.p12
PKCS12Export debug: The directory for certdb/keydb is .
PKCS12Export debug: The password file for keydb is dbpwd.txt
PKCS12Export debug: Number of user certificates = 5
PKCS12Export debug: Certificate nickname = ocspSigningCert cert-ca
PKCS12Export debug: Private key is not null
PKCS12Export debug: Certificate nickname = subsystemCert cert-ca
PKCS12Export debug: Private key is not null
PKCS12Export debug: Certificate nickname = caSigningCert cert-ca
PKCS12Export debug: Private key is not null
PKCS12Export debug: Certificate nickname = Server-Cert cert-ca
PKCS12Export debug: Private key is not null
PKCS12Export debug: Certificate nickname = auditSigningCert cert-ca
PKCS12Export debug: Private key is not null
```

第25章 REVOKER（失効要求の送信）

revoker ユーティリティーは、失効要求を CA エージェントインターフェイスに送信して、証明書を取り消します。インターフェイスにアクセスするには、失効 元は CA で許可されるサブシステムグループの一部であるエージェント証明書にアクセスできる必要があります。

revoker ツールは、以下のすべてを行うことができます。

- 16 進数のシリアル番号を一覧表示して、取り消す証明書または証明書の一覧を指定します。
- 失効理由を指定します。
- 無効な日付を指定します。
- 現在保持されている証明書の取り消しを解除します。

25.1. SYNTAX

revoker ユーティリティーの構文は以下のとおりです。

revoker -s *serialNumber* -n *rsa_nickname* [[-p *password*] | [-w *passwordFile*]] [-d *dbdir*] [-v] [-V] [-u] [-r *reasoncode*] [-i *numberOfHours*] hostname [:port]

オプション	説明
s	取り消す証明書の 16 進数でシリアル番号を指定します。たとえば、16 進数のシリアル番号は **0x31** と同様で、複数のシリアル番号を 0x **44,0x64,0x22** のようにコンマで区切って一覧表示することもできます。
n	エージェント証明書のニックネームを指定します。
p	証明書データベースのパスワードを指定します。**-w** オプションが使用されている場合は使用されません。
w	オプション:パスワードファイルへのパスを指定します。**-p** オプションが使用されている場合は使用されません。
d	オプション:セキュリティーデータベースへのパスを指定します。
v	オプション:詳細モードで操作を設定します。
V	オプション:取り消しツールのバージョンを指定し ます。
u	オプション:証明書の取り消しを解除します。つまり、証明書のステータスが on hold から active に変わります。

オプション	説明
r	証明書を取り消す理由を示します。考えられる理由を以下に示します。 ● 0 - 指定されない（デフォルト）。 ● 1- 鍵が侵害されました。 ● 2: CA キーが侵害されました。 ● 3: ユーザーの所属が変更されました。 ● 4 - 証明書が置き換えられました。 ● 5 - 操作の停止。 ● 6 - 証明書が保留中です。
i	証明書を取り消すタイミングの無効日を、現在時刻から時間単位で設定します。
hostname	要求を送信するサーバーのホスト名を指定します。DNS とネットワークの設定方法に応じて、マシン名、完全修飾ドメイン名、または IPv4 アドレスまたは IPv6 アドレスを使用できます。
port	オプション:サーバーのエージェントの SSL ポート番号を指定します。

25.2. 出力

詳細オプション(-v)を使用しないと、取り消しャー は、標準 I/O の追加出力なしで 0 の終了コードを返します。

-v オプションを指定すると、CA エージェントインターフェイスに送信された GET 要求と、返される結果(HTML ページ)が表示されます。

```
# revoker -d . -s 0x17 -n "CA Administrator of Instance pki-ca Example Domain" -p secret -v -r
6 -i 1 server.example.com:9443


GET /ca/doRevoke?
op=doRevoke&revocationReason=6&invalidityDate=1299187797000&revokeAll=(|
(certRecordId%3D0x17))&totalRecordCount=1 HTTP/1.0
port: 9443
addr='server.example.com'
family='2'
Subject: CN=server.example.com,OU=pki-ca,O=Example Domain
Issuer : CN=Certificate Authority,OU=pki-ca,O=Example Domain
-- SSL3: Server Certificate Validated.
Called mygetclientauthdata - nickname = CA Administrator of Instance pki-ca Example
Domain ID
   mygetclientauthdata - cert = 8da87b8
```

```
   mygetclientauthdata - privkey = 8de65a8
PR_Write wrote 143 bytes from bigBuf
bytes: [GET /ca/doRevoke?
op=doRevoke&revocationReason=6&invalidityDate=1299187797000&revokeAll=(|
(certRecordId%3D0x17))&totalRecordCount=1 HTTP/1.0

]
do_writes shutting down send socket
do_writes exiting with (failure = 0)
bulk cipher RC4, 128 secret key bits, 128 key bits, status: 1
connection 1 read 9000 bytes (9000 total).
these bytes read:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html
Date: Thu, 03 Mar 2011 22:29:58 GMT
Connection: close

<!-- --- BEGIN COPYRIGHT BLOCK ---
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; version 2 of the License.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

    Copyright (C) 2007 Red Hat, Inc.
    All rights reserved.
    --- END COPYRIGHT BLOCK --- -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>Revocation Result</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<SCRIPT LANGUAGE="JavaScript">
var header = new Object();
var fixed = new Object();
var recordSet = new Array;
var result = new Object();
var httpParamsCount = 0;
var httpHeadersCount = 0;
var authTokenCount = 0;
var serverAttrsCount = 0;
header.HTTP_PARAMS = new Array;
header.HTTP_HEADERS = new Array;
header.AUTH_TOKEN = new Array;
header.SERVER_ATTRS = new Array;
header.dirEnabled = "no";
```

```
header.error = null;
header.revoked = "yes";
header.totalRecordCount = 1;
var recordCount = 0;
var record;
record = new Object;
record.HTTP_PARAMS = new Array;
record.HTTP_HEADERS = new Array;
record.AUTH_TOKEN = new Array;
record.SERVER_ATTRS = new Array;
record.error=null;
record.serialNumber="17";
recordSet[recordCount++] = record;
record.recordSet = recordSet;
result.header = header;
result.fixed = fixed;
result.recordSet = recordSet;
</SCRIPT>

<BODY bgcolor="white">
<SCRIPT type="text/javascript">
//<!--
function toHex1(number)
{
   var absValue = "", sign = "";
   var digits = "0123456789abcdef";
   if (number < 0) {
      sign = "-";
      number = -number;
   }

   for(; number >= 16 ; number = Math.floor(number/16)) {
      absValue = digits.charAt(number % 16) + absValue;
   }
   absValue = digits.charAt(number % 16) + absValue;
   return sign + '0x' + absValue;
}

function toHex(number)
{
   return '0x' + number;
}

if (result.header.revoked == 'yes') {
   document.write('<font size="+1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
   document.writeln('Certificate Revocation Has Been Completed</font><br><br>');
   if (result.recordSet.length == 0 && result.header.totalRecordCount > 0) {
      document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
      document.write('All requested certificates were already revoked.');
      document.writeln('</font><br>');
   } else if (result.recordSet.length == 1) {
      if (result.recordSet[0].error == null) {
        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
```

```
        document.writeln('Certificate with serial number <b>' +
                    toHex(result.recordSet[0].serialNumber) +
                    '</b> has been revoked.');
        document.writeln('</font><br>');

        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
        if (result.header.updateCRL && result.header.updateCRL == "yes") {
           if (result.header.updateCRLSuccess != null &&
              result.header.updateCRLSuccess == "yes") {
              document.writeln('The Certificate Revocation List has been successfully
updated.');
           } else {
              document.writeln('The Certificate Revocation List update Failed');
              if (result.header.updateCRLSuccess != null)
                 document.writeln(' with error '+ result.header.updateCRLError);
              else
                 document.writeln('. No further details provided.');
           }
        } else {
           document.writeln(
              'The Certificate Revocation List will be updated '+
              'automatically at the next scheduled update.');
        }
        document.writeln('</font><br>');
/*
        if (result.header.dirEnabled != null && result.header.dirEnabled == 'yes') {
           document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica,
sans-serif">');
           if (result.header.certsUpdated > 0) {
              document.write('Directory has been successfully updated.');
           } else {
              document.write('Directory has not been updated.  See log files for more details.');
           }
           document.writeln('</font><br>');
        }
*/
      } else {
        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
        document.writeln('Certificate with serial number <b>' +
                    toHex(result.recordSet[0].serialNumber) +
                    '</b> is not revoked.<br><br>');
        document.writeln('Additional Information:');
        document.writeln('</font>');
        document.writeln('<blockquote>');
        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
        document.writeln(result.recordSet[0].error);
        document.writeln('</font>');
        document.writeln('</blockquote>');
      }
   } else if (result.recordSet.length > 1) {
      document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
      document.write('The following certificates were processed to complete revocation
```

```
request:');
      document.writeln('</font>');

      document.writeln('<blockquote>');
      document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
      var revokedCerts = 0;
      for(var i = 0; i < result.recordSet.length; i++) {
         if (result.recordSet[i].error == null) {
            revokedCerts++;
            document.writeln(toHex(result.recordSet[i].serialNumber) + ' - revoked<BR>\n');
         } else {
            document.write(toHex(result.recordSet[i].serialNumber) + ' - failed');
            if (result.recordSet[i].error != null)
               document.write(': ' + result.recordSet[i].error);
            document.writeln('<BR>\n');
         }
      }
      document.writeln('</font>');
      document.write('</blockquote>');

      if (revokedCerts > 0 && result.header.dirEnabled != null && result.header.dirEnabled ==
'yes') {
         document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
         if (result.header.updateCRL && result.header.updateCRL == "yes") {
            if (result.header.updateCRLSuccess != null &&
               result.header.updateCRLSuccess == "yes") {
               document.writeln('The Certificate Revocation List has been successfully
updated.');
            } else {
               document.writeln('The Certificate Revocation List update Failed');
               if (result.header.updateCRLSuccess != null)
                  document.writeln(' with error '+
                              result.header.updateCRLError);
               else
                  document.writeln('. No further details provided.');
               }
         } else {
            document.writeln(
               'The Certificate Revocation List will be updated '+
               'automatically at the next scheduled update.');
         }
         document.writeln('<br>');
/*
         if (result.header.certsUpdated > 0) {
            if (result.header.certsUpdated == result.header.certsToUpdate) {
               document.write('Directory has been successfully updated.');
            } else {
               document.write('Directory has been partially updated.  See log files for more
details.');
            }
         } else {
            document.write('Directory has not been updated.  See log files for more details.');
         }
*/
```

```
            document.writeln('</font><br>');
        }
    }
} else if (result.header.revoked == 'pending') {
    document.write('<font size="+1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
 document.writeln('Revocation Request Has Been Submitted</font><br><br>');
} else if (result.header.revoked == 'rejected') {
    document.write('<font size="+1" face="PrimaSans
connection 1 read 1249 bytes (10249 total).
these bytes read:
 BT, Verdana, Arial, Helvetica, sans-serif">');
    document.writeln('Certificate Revocation Has Been Rejected</font><br><br>');
    if (result.header.error != null) {
        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">Additional information:</font>');
        document.writeln('<blockquote>');
     document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
        document.writeln(result.header.error);
        document.writeln('</font>');
        document.writeln('</blockquote>');
    }
} else {
    document.write('<font size="+1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
 document.writeln('Revocation Request Cannot Be Completed</font><br><br>');
    if (result.header.error != null) {
        document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">Additional information:</font>');
        document.writeln('<blockquote>');
     document.writeln('<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-
serif">');
        document.writeln(result.header.error);
        document.writeln('</font>');
        document.writeln('</blockquote>');
    }
}
//-->
</SCRIPT>
</BODY>
</HTML>

connection 1 read 10249 bytes total. ----------------------------
```

第26章 TPSCLIENT (TPS のデバッグ)

tpsclient ツールは、TPS のデバッグまたはテストに使用できます。tpsclient は Enterprise Security Client に対応し、トークンを使用せずに、デバッグ出力を提供し、トークンの登録とフォーマットをエミュレートできます。

tpsclient ツールは、コマンド tpsclient を実行して起動します。ツールにはオプションがありません。これを実行すると、特定のコマンドを tpsclient に転送できるシェルが開きます。

```
tpsclient

Registration Authority Client
'op=help' for Help
Command>
```

tpsclient と TPS は、安全なチャンネルを確立するために一連の対称鍵に同意する必要があります。これらはいずれも相互のデフォルトトークンで設定されます。このトークンには、認証キー、Mac キー、およびキー暗号化キー(KEK)の 3 つのキーが含まれるデフォルトのキーセット(バージョン 1)が設定されます。TPS サブシステムは、デフォルトのキーセットを理解し、受け入れます。

それぞれのデフォルトのキーの値は 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4a 0x4b 0x4c 0x4d 0x4e 0x4f, 16 バイトに設定されます。デフォルト設定は、tpsclient コマンドシェル内で token_status オプションを実行することで表示されます。

```
Command>token_status
token_status
Output> life_cycle_state : '0'
Output> pin : 'password'
Output> app_ver : '00010203' (4 bytes)
Output> major_ver : '0'
Output> minor_ver : '0'
Output> cuid : '00010203040506070809' (10 bytes)
Output> msn : '00000000' (4 bytes)
Output> key_info : '0101' (2 bytes)
Output> auth_key : '404142434445464748494a4b4c4d4e4f' (16 bytes)
Output> mac_key : '404142434445464748494a4b4c4d4e4f' (16 bytes)
Output> kek_key : '404142434445464748494a4b4c4d4e4f' (16 bytes)
Result> Success - Operation 'token_status' Success (8 msec)
Command>
```

TPS が新しいマスターキーを使用するように設定されている場合、tpsclient も再設定する必要があります。そうしないと、TPS への接続を確立できません。

1. **tpsclient に入力する新しいキーセットデータを取得します。デフォルトのキーセットは TKS に保存する必要があり、マスターキーを追加する必要があります。これを行うには、TKS CS.cfg ファイルの TKS マッピングパラメーターを編集します。**

   ```
   tks.mk_mappings.#02#01=nethsm1:masterkey
   ```

   この設定は、nethsm1 トークンで masterkey という名前のマスターキー を #02#01 キーにマップするように TKS に指示します。

2. **TPS CS.cfg ファイルで update symmetric keys パラメーターを編集して、TPS で鍵のアップグレードを有効にします。**

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

この設定は、tpsclient 形式の操作中にトークンをバージョン 1 からバージョン 2 にアップグレードするように TPS に指示します。

3. 以下のように tpsclient を使用してトークンをフォーマットします。

```
tpsclient
Command>op=token_set cuid=a00192030405060708c9 app_ver=6FBBC105
key_info=0101
Command>op=token_set auth_key=40414243444546474849 4a4b4c4d4e4f
Command>op=token_set mac_key=40414243444546474849 4a4b4c4d4e4f
Command>op=token_set kek_key=40414243444546474849 4a4b4c4d4e4f

Command>op=ra_format uid=jsmith pwd=password num_threads=1
new_pin=password
```

CUID は任意の 10 バイトの文字列にすることができます。TKS が tpsclient の新しいキーセットを計算する方法に影響します。

ヒント

コマンドラインで各操作とパラメーターを入力することが面倒になる可能性があるため、入力ファイルを作成し、tpsclient コマンドをファイルを参照することができます。以下に例を示します。

```
tpsclient < /tmp/input.txt
```

例26.1「tpsclient 登録入力ファイルの例」 また、例26.2「tpsclient 形式の入力ファイルの例」 はどちらも入力ファイルの例を一覧表示します。

コマンドプロンプトは、操作中に tpsclient が指定した出力と、コマンドの最終結果を返します。

4. フォーマット操作の実行後に、tpsclient は標準出力に新しいキーセットを出力します。新しい tpsclient 入力ファイルに新しい値を保存します。入力ファイルは、実稼働 TPS サーバーで使用できます。

tpsclient は、操作のフォーマットまたは登録操作に使用できます。登録操作の入力ファイルのサンプルは 例26.1「tpsclient 登録入力ファイルの例」 に表示されます。

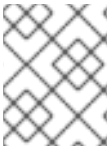
例26.1 tpsclient 登録入力ファイルの例

```
op=var_set name=ra_host value=server.example.com
op=var_set name=ra_port value=7888
op=var_set name=ra_uri value=/nk_service
op=token_set cuid=00000000000000000001
  msn=01020304 app_ver=6FBBC105 key_info=0101 major_ver=0 minor_ver=0
op=token_set auth_key=40414243444546474849 4a4b4c4d4e4f
op=token_set mac_key=40414243444546474849 4a4b4c4d4e4f
op=token_set kek_key=40414243444546474849 4a4b4c4d4e4f

op=ra_enroll uid=jdoe pwd=password new_pin=password num_threads=1
```

登録操作の入力ファイルのサンプルは 例26.2「tpsclient 形式の入力ファイルの例」 に表示されます。

例26.2 tpsclient 形式の入力ファイルの例

```
op=var_set name=ra_host value=server.example.com
op=var_set name=ra_port value=7888
op=var_set name=ra_uri value=/nk_service
op=token_set cuid=00000000000000000001
  msn=01020304 app_ver=6FBBC105 key_info=0101 major_ver=0 minor_ver=0
op=token_set auth_key=40414243444546474849 4a4b4c4d4e4f
op=token_set mac_key=40414243444546474849 4a4b4c4d4e4f
op=token_set kek_key=40414243444546474849 4a4b4c4d4e4f
op=ra_format uid=jsmith pwd=secret new_pin=newsecret num_threads=1
```



注記

ホスト用に設定されている場合、ホストの値は IPv4 アドレスまたは IPv6 アドレスになります。

26.1. SYNTAX

tpsclient ツールの構文は次のとおりです。

```
tpsclient op=operation options
```

表26.1 tpsclient 操作

操作	説明	Options
op=help	tpsclient ツールの使用とオプションをすべて一覧表示するヘルプページを表示します。	該当なし
op=debug filename=filename	デバッグを有効にします。	filename は、デバッグファイルを設定します。

操作	説明	Options
op=ra_enroll	証明書の登録をテストします。	• **UID** は、実行中のユーザーのユーザー ID を指定します。 • **pwd** は、ユーザー ID に対応するパスワードを指定します。 • **num_threads** は使用するスレッドの数を設定します。 • **secureid_pin** は、トークンパスワードを提供します。 • **keygen** は、サーバー側の鍵生成を有効にするかどうかを設定します。
op=ra_reset_pin	トークン PIN をリセットします。	• **UID** は、実行中のユーザーのユーザー ID を指定します。 • **pwd** は、ユーザー ID に対応するパスワードを指定します。 • **num_threads** は使用するスレッドの数を設定します。 • **secureid_pin** は、トークンパスワードを提供します。 • **new_pin** は、新しい PIN（トークンパスワード）を設定します。
op=ra_update	アプレットを更新します。	• **UID** は、実行中のユーザーのユーザー ID を指定します。 • **pwd** は、ユーザー ID に対応するパスワードを指定します。 • **num_threads** は使用するスレッドの数を設定します。 • **secureid_pin** は、トークンパスワードを提供します。

操作	説明	Options
op=token_set	トークン値を設定します。	この操作の使用方法は name=value で、トークン名と説明を設定します。
op=token_status	現在のトークンステータス/を返します。	該当なし
op=var_get	変数の現在の値を取得します。	これには、使用量の **name=**name があります。name はチェックされる変数です。
op=var_list	考えられるすべての変数を一覧表示します。	該当なし
op=var_set	変数値を設定します。	• **name** は変数の名前を設定します。 • **値** は名前付き変数の値を設定します。

第27章 KRATOOL （秘密鍵のラップ）

一部の秘密鍵（主に古いデプロイメントでは）は、Key Recovery Authority (KRA)でアーカイブされた場合に SHA-1、1024 ビットのストレージキーでラップされました。プロセッサーの速度とアルゴリズムが破損しているため、このアルゴリズムはセキュリティーレベルが低くなります。セキュリティー対策として、新しい強力なストレージ鍵 (SHA-256、2048 ビット鍵) で秘密鍵を再ラップできます。

注記

KRATool ユーティリティーは、1つの KRA から秘密鍵をエクスポートし、新しいストレージキーで再ラップしてから、新しい KRA にインポートできるため、複数の KRA インスタンスを 1つの KRA インスタンスに統合するプロセスの一部として使用できます。

27.1. SYNTAX

KRATool ユーティリティーを実行して、キーの再ラップ、キーの再番号、またはその両方を実行できます。

キーを再ラップするための構文：

```
KRATool -kratool_config_file /path/to/tool_config_file
 -source_ldif_file /path/to/original_ldif_file
 -target_ldif_file /path/to/newinstance_ldif_file
 -log_file /path/to/tool_log_file
 [-source_pki_security_database_path /path/to/nss_databases
 -source_storage_token_name /path/to/token
 -source_storage_certificate_nickname storage_certificate_nickname
 -target_storage_certificate_file /path/to/new_ASCII_storage_cert
 [-source_pki_security_database_pwdfile /path/to/password_file]]
 [-source_kra_naming_context name -target_kra_naming_context name]
 [-process_requests_and_key_records_only]
```

キーを再数値する構文：

```
KRATool -kratool_config_file /path/to/tool_config_file
 -source_ldif_file /path/to/original_ldif_file
 -target_ldif_file /path/to/newinstance_ldif_file
 -log_file /path/to/tool_log_file
 [-append_id_offset prefix_to_add | -remove_id_offset prefix_to_remove]
 [-source_kra_naming_context name -target_kra_naming_context name]
 [-process_requests_and_key_records_only]
```

オプション	説明
必須パラメーター	

オプション	説明
-kratool_config_file	ツールが使用する設定ファイルの完全パスおよびファイル名を指定します。この設定プロセスは、既存のキーレコードで特定のパラメーターを処理する方法、フォーマットの変更（命名コンテキストの変更やオフセットの追加など）、または変更日を更新するかどうかをツールに指示します。設定ファイルが必要で、デフォルトの ファイルが ツールに含まれています。ファイル形式については、「.cfg File」で説明されています。
-source_ldif_file	古い KRA からのキーデータをすべて含む LDIF ファイルの完全なパスおよびファイル名を指定します。
-target_ldif_file	ツールが 新しい KRA からすべてのキーデータを書き込む LDIF ファイルの完全なパスおよびファイル名を指定します。このファイルは、実行中のツールにより作成されます。
-log_file	ツールの進捗とメッセージをログに記録するために使用するログファイルのパスとファイル名を指定します。このファイルは、実行中のツールにより作成されます。
オプションのパラメーター	
-source_kra_naming_context	元の KRA インスタンスの命名コンテキスト（元の KRA を参照する DN 要素）を指定します。キー関連の LDIF エントリーには、cn=1,ou=kra,ou=requests,dc=alpha.example.com-pki-kra などの KRA インスタンス名を持つ DN があります。そのエントリーの命名コンテキストは、DN 値 alpha.example.com-pki-kra です。これらのエントリーは、古い KRA インスタンスの命名コンテキストから新しい KRA インスタンスの命名コンテキストに、自動的に名前を変更できます。 この引数はオプションですが、ターゲット KRA にインポートする前に LDIF ファイルを編集する必要がないため、推奨されます。 この引数を使用する場合は、**-target_kra_naming_context** 引数も使用する必要があります。

オプション	説明
-target_kra_naming_context	新しい KRA インスタンスの命名コンテキストを指定します。元のキーエントリーを に変更 する 必要がある名前です。キー関連の LDIF エントリーには、cn=1,ou=kra,ou=requests,dc=omega.example.com-pki-kra などの KRA インスタンス名を持つ DN があります。そのエントリーの命名コンテキストは、DN 値 omega.example.com-pki-kra です。これらのエントリーは、古い KRA インスタンスから新しい KRA インスタンスの命名コンテキストに、自動的に名前を変更できます。 この引数はオプションですが、ターゲット KRA にインポートする前に LDIF ファイルを編集する必要がないため、推奨されます。 この引数を使用する場合は、**-source_kra_naming_context** 引数も使用する必要があります。
-process_requests_and_key_records_only	ソース LDIF ファイルから設定エントリーを削除し、キーと要求エントリーのみを残します。 この引数はオプションですが、ターゲット KRA にインポートする前に LDIF ファイルを編集する必要がないため、推奨されます。
rewrap パラメーター	
-source_pki_security_database_path	古い KRA インスタンスによって使用される NSS セキュリティーデータベースが含まれるディレクトリーへの完全パスを指定します。 このオプションは、他の rewrap パラメーターが使用される場合に必要です。
-source_storage_token_name	内部トークンの内部 キー ストレージトークンや、ハードウェアトークン名として NHSM6000-OCS などの KRA データを格納するトークンの名前を指定します。 このオプションは、他の rewrap パラメーターが使用される場合に必要です。
-source_storage_certificate_nickname	古い KRA インスタンスの KRA ストレージ証明書のニックネームを指定します。この証明書は 古い KRA インスタンスのセキュリティーデータベースに配置されるか、セキュリティーデータベースにはハードウェアトークンの証明書へのポインターが含まれます。 このオプションは、他の rewrap パラメーターが使用される場合に必要です。

オプション	説明
-target_storage_certificate_file	新しい KRA インスタンスのストレージ証明書の ASCII 形式のファイルのパスおよびファイル名を指定します。ストレージ証明書は新しい KRA のデータベースからエクスポートし、**KRATool** を実行する前にアクセス可能な場所に保存する必要があります。 このオプションは、他の rewrap パラメーターが使用される場合に必要です。
-source_pki_security_database_pwdfile	**-source_storage_token_name** オプションで指定したストレージトークンのパスワードのみが含まれるパスワードファイルへのパスとファイル名を指定します。 この引数は、他の rewrap パラメーターが使用される場合は任意です。この引数を使用しない場合、スクリプトはパスワードを要求します。
番号オフセットパラメーター	
-append_id_offset	競合の可能性を防ぐために、インポートされたすべてのキーに先行される ID 番号を指定します。**KRATool** を使用してエクスポートされたキーを持つすべての KRA インスタンスに一意の ID オフセットを使用する必要があります。 **-append_id_offset** を使用する場合は、**-remove_id_offset** オプションを使用し ない でください。
-remove_id_offset	インポートされたすべてのキーの先頭から削除する ID 番号を指定します。 **-remove_id_offset** を使用する場合は、**-append_id_offset** オプションを使用し ない でください。

27.2. .CFG FILE

必要な設定ファイルは、LDIF ファイルのキーアーカイブおよびキー要求エントリーで属性を処理する方法を **KRATool** に指示します。エントリーには 6 つのタイプがあります。

- CA 登録要求
- TPS 登録要求
- CA キーレコード
- TPS キーレコード
- CA および TPS リカバリー要求(KRA で同じ処理)

各キーおよびキー要求には、その種類のレコードに固有の属性を含む LDAP エントリーがあります。たとえば、リカバリー要求の場合は以下のようになります。

```
dn: cn=1,ou=kra,ou=requests,dc=alpha.example.com-pki-kra
objectClass: top
objectClass: request
objectClass: extensibleObject
requestId: 011
requestState: complete
dateOfCreate: 20110121181006Z
dateOfModify: 20110524094652Z
extdata-kra--005ftrans--005fdeskey: 3#C7#82#0F#5D#97GqY#0Aib#966#E5B#F56#F24n#
 F#9E#98#B3
extdata-public--005fkey:
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDu6E3uG+Ep27bF1
 yTWvwIDAQAB
extdata-archive: true
extdata-requesttype: netkeyKeygen
extdata-iv--005fs: %F2%67%45%96%41%D7%FF%10
extdata-requestversion: 8.1.0
extdata-requestortype: NETKEY_RA
extdata-keyrecord: 1
extdata-wrappeduserprivate:
%94%C1%36%D3%EA%4E%36%B5%42%91%AB%47%34%C0%35%A3%6
 F%E8%10%A9%B1%25%F4%BE%9C%11%D1%B3%3D%90%AB%79
extdata-userid: jmagne
extdata-keysize: 1024
extdata-updatedby: TPS-alpha.example.com-7889
extdata-dbstatus: UPDATED
extdata-cuid: 40906145C76224192D2B
extdata-requeststatus: complete
extdata-requestid: 1
extdata-result: 1
requestType: netkeyKeygen
cn: 1
creatorsName: cn=directory manager
modifiersName: cn=directory manager
createTimestamp: 20110122021010Z
modifyTimestamp: 20110122021010Z
nsUniqueId: b2891805-1dd111b2-a6d7e85f-2c2f0000
```

その情報の多くは、スクリプト処理を変更せずに渡すため、新しいターゲット KRA にだけ入力されます。ただし、これらの属性の一部は、新しい KRA インスタンスに一致するように変更されている CN および DN など、編集可能であり、編集する必要があります。安全に変更できるフィールドは、それぞれのタイプのキーエントリーの設定ファイルに一覧表示されています。（リストされていない属性は、どの状況でもツールによってアクセスされません。）

フィールドを編集する 必要 がある場合は、ツールがレコード ID 番号を更新したり、エントリーの名前を変更したりできます。これにより、設定ファイルで値が true に設定されます。たとえば、以下の設定は、すべての CA 登録要求の CN、DN、ID 番号、最終変更日、および関連するエントリーノートを更新します。

```
kratool.ldif.caEnrollmentRequest.cn=true
kratool.ldif.caEnrollmentRequest.dateOfModify=true
kratool.ldif.caEnrollmentRequest.dn=true
kratool.ldif.caEnrollmentRequest.extdata.keyRecord=true
kratool.ldif.caEnrollmentRequest.extdata.requestNotes=true
kratool.ldif.caEnrollmentRequest.requestId=true
```

行が true に設定されている場合、属性は LDIF ファイルで処理されます。デフォルトでは、可能な属性がすべて処理されます。行を false に設定すると、KRATool はその属性を省略し、値を変更せずに渡します。たとえば、これにより最後に変更した時刻が変更され、KRATool の実行時に更新されないようになります。

```
kratool.ldif.caEnrollmentRequest.dateOfModify=false
```

ヒント

キーの登録、レコード、および要求には、管理者がプロセスに関するメモを入力できるオプションの notes 属性があります。KRATool が実行されると、その属性にメモを追加するか、実行中のツールに関する情報、実行された操作、およびタイムスタンプが含まれる 属性を追加します。

```
extdata-requestnotes: [20110701150056Z]: REWRAPPED the 'existing DES3 symmetri
 c session key' with the '2048-bit RSA public key' obtained from the target s
 torage certificate + APPENDED ID offset '100000000000' + RENAMED source KRA
 naming context 'alpha.example.com-pki-kra' to target KRA naming context 'ome
 ga.example.com-pki-kra' + PROCESSED requests and key records ONLY!
```

この情報は、KRA の監査とメンテナーンスの両方に非常に便利です。そのため、true に設定されたすべてのキーレコードタイプの *extdata.requestNotes* パラメーターを保持すると便利です。

重要

デフォルトの kratool.cfg のすべてのパラメーター行は、ツールの呼び出し時に使用する .cfg ファイルに存在している 必要があり ます。行は省略できず、すべての行には有効な値が必要です(true または false)。ファイルが適切にフォーマットされていない場合、KRATool は失敗します。

.cfg ファイルのフォーマットは、インスタンスの CS.cfg ファイルで使用されるフォーマットと同じです。

デフォルトの .cfg ファイルは KRATool スクリプトに含まれています。このファイル(例27.1「デフォルトの kratool.cfg ファイル」)は、カスタムファイルにコピーして編集するか、直接編集してツールで使用できます。

例27.1 デフォルトの kratool.cfg ファイル

```
kratool.ldif.caEnrollmentRequest._000=##########################################
kratool.ldif.caEnrollmentRequest._001=##     KRA CA Enrollment Request      ##
kratool.ldif.caEnrollmentRequest._002=##########################################
kratool.ldif.caEnrollmentRequest._003=##                                    ##
kratool.ldif.caEnrollmentRequest._004=##  NEVER allow 'KRATOOL' the ability ##
kratool.ldif.caEnrollmentRequest._005=##  to change the CA 'naming context' ##
kratool.ldif.caEnrollmentRequest._006=##  data in the following fields:     ##
kratool.ldif.caEnrollmentRequest._007=##                                    ##
kratool.ldif.caEnrollmentRequest._008=##    extdata-auth--005ftoken;uid     ##
kratool.ldif.caEnrollmentRequest._009=##    extdata-auth--005ftoken;userid  ##
kratool.ldif.caEnrollmentRequest._010=##    extdata-updatedby               ##
kratool.ldif.caEnrollmentRequest._011=##                                    ##
```

```
kratool.ldif.caEnrollmentRequest._012=##  NEVER allow 'KRATOOL' the ability ##
kratool.ldif.caEnrollmentRequest._013=##  to change CA 'numeric' data in    ##
kratool.ldif.caEnrollmentRequest._014=##  the following fields:            ##
kratool.ldif.caEnrollmentRequest._015=##                                   ##
kratool.ldif.caEnrollmentRequest._016=##    extdata-requestId              ##
kratool.ldif.caEnrollmentRequest._017=##                                   ##
kratool.ldif.caEnrollmentRequest._018=########################################
kratool.ldif.caEnrollmentRequest.cn=true
kratool.ldif.caEnrollmentRequest.dateOfModify=true
kratool.ldif.caEnrollmentRequest.dn=true
kratool.ldif.caEnrollmentRequest.extdata.keyRecord=true
kratool.ldif.caEnrollmentRequest.extdata.requestNotes=true
kratool.ldif.caEnrollmentRequest.requestId=true
kratool.ldif.caKeyRecord._000=########################################
kratool.ldif.caKeyRecord._001=##         KRA CA Key Record          ##
kratool.ldif.caKeyRecord._002=########################################
kratool.ldif.caKeyRecord._003=##                                    ##
kratool.ldif.caKeyRecord._004=##  NEVER allow 'KRATOOL' the ability  ##
kratool.ldif.caKeyRecord._005=##  to change the CA 'naming context'  ##
kratool.ldif.caKeyRecord._006=##  data in the following fields:      ##
kratool.ldif.caKeyRecord._007=##                                    ##
kratool.ldif.caKeyRecord._008=##    archivedBy                      ##
kratool.ldif.caKeyRecord._009=##                                    ##
kratool.ldif.caKeyRecord._010=########################################
kratool.ldif.caKeyRecord.cn=true
kratool.ldif.caKeyRecord.dateOfModify=true
kratool.ldif.caKeyRecord.dn=true
kratool.ldif.caKeyRecord.privateKeyData=true
kratool.ldif.caKeyRecord.serialno=true
kratool.ldif.namingContext._000=##############################################
kratool.ldif.namingContext._001=##       KRA Naming Context Fields        ##
kratool.ldif.namingContext._002=##############################################
kratool.ldif.namingContext._003=##                                        ##
kratool.ldif.namingContext._004=##  NEVER allow 'KRATOOL' the ability to  ##
kratool.ldif.namingContext._005=##  change the CA 'naming context' data   ##
kratool.ldif.namingContext._006=##  in the following 'non-KeyRecord /     ##
kratool.ldif.namingContext._007=##  non-Request' fields (as these records ##
kratool.ldif.namingContext._008=##  should be removed via the option to   ##
kratool.ldif.namingContext._009=##  process requests and key records only ##
kratool.ldif.namingContext._010=##  if this is a KRA migration):          ##
kratool.ldif.namingContext._011=##                                        ##
kratool.ldif.namingContext._012=##    cn                                  ##
kratool.ldif.namingContext._013=##    sn                                  ##
kratool.ldif.namingContext._014=##    uid                                 ##
kratool.ldif.namingContext._015=##    uniqueMember                      ##
kratool.ldif.namingContext._016=##                                        ##
kratool.ldif.namingContext._017=##  NEVER allow 'KRATOOL' the ability to  ##
kratool.ldif.namingContext._018=##  change the KRA 'naming context' data  ##
kratool.ldif.namingContext._019=##  in the following 'non-KeyRecord /     ##
kratool.ldif.namingContext._020=##  non-Request' fields (as these records ##
kratool.ldif.namingContext._021=##  should be removed via the option to   ##
kratool.ldif.namingContext._022=##  process requests and key records only ##
kratool.ldif.namingContext._023=##  if this is a KRA migration):          ##
kratool.ldif.namingContext._024=##                                        ##
kratool.ldif.namingContext._025=##     dc                                 ##
kratool.ldif.namingContext._026=##     dn                                 ##
```

```
kratool.ldif.namingContext._027=##     uniqueMember                  ##
kratool.ldif.namingContext._028=##                                   ##
kratool.ldif.namingContext._029=##  NEVER allow 'KRATOOL' the ability to  ##
kratool.ldif.namingContext._030=##  change the TPS 'naming context' data  ##
kratool.ldif.namingContext._031=##  in the following 'non-KeyRecord /     ##
kratool.ldif.namingContext._032=##  non-Request' fields (as these records ##
kratool.ldif.namingContext._033=##  should be removed via the option to   ##
kratool.ldif.namingContext._034=##  process requests and key records only ##
kratool.ldif.namingContext._035=##  if this is a KRA migration):         ##
kratool.ldif.namingContext._036=##                                   ##
kratool.ldif.namingContext._037=##   uid                             ##
kratool.ldif.namingContext._038=##   uniqueMember                    ##
kratool.ldif.namingContext._039=##                                   ##
kratool.ldif.namingContext._040=##  If '-source_naming_context           ##
kratool.ldif.namingContext._041=##  original source KRA naming context'   ##
kratool.ldif.namingContext._042=##  and '-target_naming_context          ##
kratool.ldif.namingContext._043=##  renamed target KRA naming context'    ##
kratool.ldif.namingContext._044=##  options are specified, ALWAYS         ##
kratool.ldif.namingContext._045=##  require 'KRATOOL' to change the       ##
kratool.ldif.namingContext._046=##  KRA 'naming context' data in ALL of   ##
kratool.ldif.namingContext._047=##  the following fields in EACH of the   ##
kratool.ldif.namingContext._048=##  following types of records:          ##
kratool.ldif.namingContext._049=##                                   ##
kratool.ldif.namingContext._050=##   caEnrollmentRequest:               ##
kratool.ldif.namingContext._051=##                                   ##
kratool.ldif.namingContext._052=##    dn                              ##
kratool.ldif.namingContext._053=##    extdata-auth--005ftoken;user     ##
kratool.ldif.namingContext._054=##    extdata-auth--005ftoken;userdn    ##
kratool.ldif.namingContext._055=##                                   ##
kratool.ldif.namingContext._056=##   caKeyRecord:                    ##
kratool.ldif.namingContext._057=##                                   ##
kratool.ldif.namingContext._058=##    dn                              ##
kratool.ldif.namingContext._059=##                                   ##
kratool.ldif.namingContext._060=##   recoveryRequest:                 ##
kratool.ldif.namingContext._061=##                                   ##
kratool.ldif.namingContext._062=##    dn                              ##
kratool.ldif.namingContext._063=##                                   ##
kratool.ldif.namingContext._064=##   tpsKeyRecord:                   ##
kratool.ldif.namingContext._065=##                                   ##
kratool.ldif.namingContext._066=##    dn                              ##
kratool.ldif.namingContext._067=##                                   ##
kratool.ldif.namingContext._068=##   tpsNetkeyKeygenRequest:          ##
kratool.ldif.namingContext._069=##                                   ##
kratool.ldif.namingContext._070=##    dn                              ##
kratool.ldif.namingContext._071=##                                   ##
kratool.ldif.namingContext._072=##############################################
kratool.ldif.recoveryRequest._000=#######################################
kratool.ldif.recoveryRequest._001=##  KRA CA / TPS Recovery Request  ##
kratool.ldif.recoveryRequest._002=#######################################
kratool.ldif.recoveryRequest.cn=true
kratool.ldif.recoveryRequest.dateOfModify=true
kratool.ldif.recoveryRequest.dn=true
kratool.ldif.recoveryRequest.extdata.requestId=true
kratool.ldif.recoveryRequest.extdata.requestNotes=true
kratool.ldif.recoveryRequest.extdata.serialnumber=true
kratool.ldif.recoveryRequest.requestId=true
```

```
kratool.ldif.tpsKeyRecord._000=##########################################
kratool.ldif.tpsKeyRecord._001=##        KRA TPS Key Record         ##
kratool.ldif.tpsKeyRecord._002=##########################################
kratool.ldif.tpsKeyRecord._003=##                                   ##
kratool.ldif.tpsKeyRecord._004=##  NEVER allow 'KRATOOL' the ability  ##
kratool.ldif.tpsKeyRecord._005=##  to change the TPS 'naming context' ##
kratool.ldif.tpsKeyRecord._006=##  data in the following fields:      ##
kratool.ldif.tpsKeyRecord._007=##                                   ##
kratool.ldif.tpsKeyRecord._008=##    archivedBy                     ##
kratool.ldif.tpsKeyRecord._009=##                                   ##
kratool.ldif.tpsKeyRecord._010=##########################################
kratool.ldif.tpsKeyRecord.cn=true
kratool.ldif.tpsKeyRecord.dateOfModify=true
kratool.ldif.tpsKeyRecord.dn=true
kratool.ldif.tpsKeyRecord.privateKeyData=true
kratool.ldif.tpsKeyRecord.serialno=true
kratool.ldif.tpsNetkeyKeygenRequest._000=####################################
kratool.ldif.tpsNetkeyKeygenRequest._001=##  KRA TPS Netkey Keygen Request  ##
kratool.ldif.tpsNetkeyKeygenRequest._002=####################################
kratool.ldif.tpsNetkeyKeygenRequest._003=##                              ##
kratool.ldif.tpsNetkeyKeygenRequest._004=##  NEVER allow 'KRATOOL' the      ##
kratool.ldif.tpsNetkeyKeygenRequest._005=##  ability to change the          ##
kratool.ldif.tpsNetkeyKeygenRequest._006=##  TPS 'naming context' data in   ##
kratool.ldif.tpsNetkeyKeygenRequest._007=##  the following fields:          ##
kratool.ldif.tpsNetkeyKeygenRequest._008=##                              ##
kratool.ldif.tpsNetkeyKeygenRequest._009=##    extdata-updatedby            ##
kratool.ldif.tpsNetkeyKeygenRequest._010=##                              ##
kratool.ldif.tpsNetkeyKeygenRequest._011=####################################
kratool.ldif.tpsNetkeyKeygenRequest.cn=true
kratool.ldif.tpsNetkeyKeygenRequest.dateOfModify=true
kratool.ldif.tpsNetkeyKeygenRequest.dn=true
kratool.ldif.tpsNetkeyKeygenRequest.extdata.keyRecord=true
kratool.ldif.tpsNetkeyKeygenRequest.extdata.requestId=true
kratool.ldif.tpsNetkeyKeygenRequest.extdata.requestNotes=true
kratool.ldif.tpsNetkeyKeygenRequest.requestId=true
```

27.3. 例

KRATool は、2 つの操作を実行します。新しい秘密鍵でキーを再ラップでき、登録やリカバリー要求など、キーレコードの LDIF ファイルエントリーで属性を再配列できます。少なくとも 1 つの操作 (再ラップまたは再番号付け) を実行する必要があり、両方を 1 回の呼び出しで実行できます。

例27.2 キーの再ラップ

キーを再ラップする場合、ツールはソース KRA とそのストレージ証明書の 元 の NSS データベースにアクセスして、キーをアンラップするだけでなく、新しい KRA のストレージ証明書（キーの再ラップに使用される）にアクセスできる必要があります。

```
KRATool -kratool_config_file "/usr/share/pki/java-tools/KRATool.cfg" -source_ldif_file
"/tmp/files/originalKRA.ldif" -target_ldif_file "/tmp/files/newKRA.ldif" -log_file
"/tmp/kratool.log" -source_pki_security_database_path "/tmp/files/" -
source_storage_token_name "Internal Key Storage Token"  -
source_storage_certificate_nickname "storageCert cert-pki-kra"  -target_storage_certificate_file
"/tmp/files/omega.cert"
```

例27.3 キーの番号変更

複数の KRA インスタンスが単一のインスタンスにマージされる場合は、キーまたは要求レコードに競合する CN、DN、シリアル番号、または要求 ID 番号がないことを確認することが重要です。これらの値を処理して、既存の値に新しい大きな数値を追加できます。

CN では、新しい番号は元の CN の追加と追加された番号になります。たとえば、CN が 4 で、追加番号が 1000000 の場合、新しい CN は 1000004 になります。

シリアル番号および要求 ID の場合、値は常に数字のカウントと値になります。そのため、CN の 4 のシリアル番号は 014、または 1 桁の数字と CN 値を持ちます。追加番号が 1000000 の場合、新しいシリアル番号は 071000004 になります(7 桁の数字)。次に、追加番号(1000000)と元の値(4)の合計になります。

```
KRATool -kratool_config_file "/usr/share/pki/java-tools/KRATool.cfg" -source_ldif_file
"/tmp/files/originalKRA.ldif" -target_ldif_file "/tmp/files/newKRA.ldif" -log_file
"/tmp/kratool.log" -append_id_offset 100000000000
```

例27.4 元の番号の復元

例27.3「キーの番号変更」 のように、キーエントリーに番号が追加された場合には、その数字を削除することもできます。CN の更新に加えて、シリアル番号や要求 ID 番号などの関連する番号も再構築します。元の数が競合を防ぐのに十分な大きさではない場合や、移行または KRA 統合プロセスのテストの一環として、再番号アクションを元に戻す必要がある場合があります。

```
KRATool -kratool_config_file "/usr/share/pki/java-tools/KRATool.cfg" -source_ldif_file
"/tmp/files/originalKRA.ldif" -target_ldif_file "/tmp/files/newKRA.ldif" -log_file
"/tmp/kratool.log" -remove_id_offset 100000000000
```

例27.5 1つのコマンドでの番号変更と再ラップ

同じ呼び出しで再ラップおよび番号変更操作を実行できます。

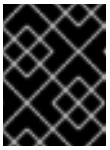
```
KRATool -kratool_config_file "/usr/share/pki/java-tools/KRATool.cfg" -source_ldif_file
"/tmp/files/originalKRA.ldif" -target_ldif_file "/tmp/files/newKRA.ldif" -log_file
"/tmp/kratool.log" -source_pki_security_database_path "/tmp/files/" -
source_storage_token_name "Internal Key Storage Token" -
source_storage_certificate_nickname "storageCert cert-pki-kra" -
target_storage_certificate_file "/tmp/files/omega.cert" -append_id_offset 100000000000
```

27.4. 使用方法

この手順では、1つの Certificate System 7.1 KRA に保存されているキーを再ラップし、Certificate System 8.1 KRA に保存します。これが唯一のユースケースではありません。このツールは、ソースとターゲットの両方と同じインスタンスで実行して既存のキーを再ラップすることも、キーをまったく再ラップせずに複数の KRA インスタンスから単一のインスタンスにキーをコピーするために使用することもできます。

1. 新しい KRA インスタンスおよびマシンを準備します。

   a. 新しい Red Hat Certificate System 8.1 KRA インスタンスをインストールして設定します。

   重要

   ストレージキーのサイズとタイプを 2048 ビットおよび RSA に設定します。

   b. 新しい KRA を停止します。

   ```
   [root@newkra ~]# service pki-kra stop
   ```

   c. 古い KRA からエクスポートしたキーデータを保存するデータディレクトリーを作成します。

   ```
   [root@newkra ~]# mkdir -p /export/pki
   ```

   d. 新しい KRA のパブリックストレージ証明書を、新しいデータディレクトリーのフラットファイルにエクスポートします。

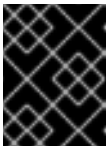
   ```
   [root@newkra ~]# certutil -L -d /var/lib/pki-kra/alias/ -n "storageCert cert-pki-kra" -a > /export/pki/newKRA.cert
   ```

   e. 同じマシンにある場合は、新しい KRA の Directory Server インスタンスを停止します。

   ```
   >[root@newkra ~]# service dirsrv stop
   ```

   f. 新しい KRA の設定情報をエクスポートします。

   ```
   [root@newkra ~]# /usr/lib[64]/disrv/slapd-instanceName/db2ldif -n newkra.example.com-pki-kra -a /export/pki/newkra.ldif
   ```

   重要

   LDIF ファイルに、末尾に空白行が 1 つ含まれていることを確認してください。

2. 古い KRA インスタンスからキーデータをエクスポートおよび準備します。

   a. エクスポートされたキーデータを保存するデータディレクトリーを作成します。

   ```
   [root@oldkra ~]# mkdir -p /export/pki
   ```

   b. [root@oldkra ~]# db2ldif などのツールを使用して、元の KRA インスタンスから情報をエクスポートします。これは、移行 ガイドの KRA の章の 7.1 から 8.1 への移行手順の一部として行われます。

   c. エクスポートしたデータの LDIF をデータディレクトリーにコピーし、アーカイブ CA を変更するためにデータファイルを更新します。

```
[root@oldkra ~]# cp /path/to/rhcs80-pki-kra.ldif /export/pki

[root@oldkra ~]# sed -i -e "s/^archivedBy: kra_trusted_agent/archivedBy: CA/g"
alpha.ldif
```

d. マシン上のすべての Certificate System サーバーを停止します。

e. NSS データベースをデータディレクトリーにコピーします。たとえば、7.1 KRA の場合は以下のようになります。

```
[root@oldkra ~]# cp -p /opt/redhat-cs/alias/cert-instance-kra-cert8.db
/export/pki/cert8.db

[root@oldkra ~]# cp -p /opt/redhat-cs/alias/cert-instance-kra-key3.db
/export/pki/key3.db

[root@oldkra ~]# cp -p /opt/redhat-cs/alias/secmod.db /export/pki/secmod.db
```

f. KRATool を古い KRA インスタンスを持つマシンにコピーし、そのすべての依存関係をプルします。7.x システムの場合は、nsutil.jar ファイルおよび cmsutil.jar ファイルを含めます（これらのファイルは 8.0 システムですでに利用可能です）。以下に例を示します。

```
[root@oldkra ~]# mkdir -p /usr/share/pki/java-tools

[root@oldkra ~]# mkdir -p /usr/share/java/pki

[root@oldkra ~]# cd /usr/share/java/pki

[root@oldkra ~]# sftp root@newkra.example.com
sftp> cd /usr/share/java/pki
sftp> get nsutil.jar
sftp> get cmsutil.jar
sftp> get cstools.jar
sftp> lcd /usr/share/pki/java-tools
sftp> cd /usr/share/pki/java-tools
sftp> get KRATool.cfg
sftp> lcd /usr/bin
sftp> cd /usr/bin
sftp> get KRATool
sftp> quit
```

重要

マシンには Java 1.6.0 がインストールされている必要があります。

g. 7.1 の KRA の場合。古い ldapjdk.jar ファイルから新しい 8.x の場所へのシンボリックリンクを作成します。

```
[root@oldkra ~]# ln -s /opt/redhat-cs/bin/cert/jars/ldapjdk.jar
/usr/share/java/ldapjdk.jar
```

h. データディレクトリーを開きます。

```
[root@oldkra ~]#cd /export/pki
```

i. パブリックストレージキーを持つファイルを新しい KRA マシンから古い KRA マシンにコピーします。以下に例を示します。

```
[root@oldkra ~]# sftp root@newkra.example.com
sftp> cd /export/pki
sftp> get newKRA.cert
sftp> quit
```

j. 必要に応じて、ツールで使用するデフォルトの KRATool.cfg ファイルを編集します。デフォルトのファイルは、変更せずに使用することもできます。

k. KRATool を実行します。これらのパラメーターはすべて 1行に指定する必要があります。

```
[root@oldkra ~]# KRATool -kratool_config_file "/usr/share/pki/java-
tools/KRATool.cfg"
        -source_ldif_file /export/pki/rhcs80-pki-kra.ldif
        -target_ldif_file /export/pki/old2newKRA.ldif
        -log_file /export/pki/kratool.log
        -source_pki_security_database_path /export/pki
        -source_storage_token_name 'Internal Key Storage Token'
        -source_storage_certificate_nickname 'storageCert cert-pki-kra'
        -target_storage_certificate_file /export/pki/newKRA.cert
        -append_id_offset 100000000000
        -source_kra_naming_context "oldkra.example.com-pki-kra"
        -target_kra_naming_context "newkra.example.com-pki-kra"
        -process_requests_and_key_records_only
```

このコマンドは、元のデータベースに保存されているトークンへのパスワードを要求します。

これが完了すると、コマンドは -target_ldif_file,old2newKRA.ldif で指定されたファイルを作成します。

l. LDIF ファイルを新しい KRA マシンにコピーします。以下に例を示します。

```
[root@oldkra ~]# scp /export/pki/old2newKRA.ldif
root@newkra.example.com:/export/pki
```

重要

LDIF ファイルに、末尾に空白行が1つ含まれていることを確認してください。

3. 複数の KRA インスタンスがマージされている場合は、それらのデータを 1つのインポート操作にマージできます。マージされるすべての KRA に対して手順 2 を実行します。

個別の LDIF ファイルを作成するには -target_ldif_file の一意の値を指定し、LDIF ファイルが連結されるときに競合が発生しないように、一意の -append_id_offset 値を指定します。

4. 新しい KRA マシンで、古いキーデータを含む LDIF ファイルをインポートします。

a. データディレクトリーを開きます。

```
[root@newkra ~]# cd /export/pki
```

b. 新しい KRA 設定 LDIF ファイルと、古い KRA インスタンス用にエクスポートされたすべての LDIF を連結します。以下に例を示します。

```
[root@newkra ~]# cat newkra.ldif old2newKRA.ldif > combined.ldif
```

c. LDIF を Certificate System 8.1 KRA インスタンスの Directory Server データベースにインポートします。

```
[root@newkra ~]# /usr/lib[64]/disrv/slapd-instanceName/ldif2db -n
newkra.example.com-pki-kra -i /export/pki/combined.ldif
```

d. 新しい KRA の Directory Server インスタンスを起動します。

```
[root@newkra ~]# service dirsrv start
```

e. 新しい KRA を起動します。

```
[root@newkra ~]# service pki-kra start
```

索引

A

C

E

G

K

付録A 更新履歴

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 9.0-1 **Wed Sep 16 2015** **Aneta Petrová**
共有および非共有インスタンスのドキュメントを含む非同期更新

改訂 9.0-0 **Fri Aug 28 2015** **Tomáš Čapek**
Red Hat Certificate System 9 リリースのバージョン