Red Hat Enterprise Linux 6

シングルサインオンおよびスマートカードの管理

エンタープライズセキュリティークライアントの使用

Last Updated: 2020-07-17

Red Hat Enterprise Linux 6 シングルサインオンおよびスマートカードの管理

エンタープライズセキュリティークライアントの使用

Aneta Šteflová Petrová
Red Hat Customer Content Services
aneta@redhat.com

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律上の通知

概要

本ガイドは、Red Hat Enterprise Linux 6 のユーザーおよび管理者向けに、Enterprise Security Client を使用して個人証明書および鍵を管理する方法について記載しています。Enterprise Security Client は、Red Hat Certificate System トークン管理システムのフロントエンドとして機能すシンプルな GUI です。Enterprise Security Client を使用することで、Red Hat Enterprise Linux 6 のユーザーは、シングルサインオンソリューションの一部として、スマートカードのフォーマットと管理が簡単になります。

目次

第1章 ENTERPRISE SECURITY CLIENT の概要

Enterprise Security Client は、スマートカードの管理を簡素化する Red Hat Certificate System のツールです。エンドユーザーは、セキュリティートークン (smart カード) を使用して、シングルサインオンアクセスやクライアント認証などのアプリケーションに使用されるユーザー証明書を格納できます。エンドユーザーには、署名、暗号化、およびその他の暗号化機能に必要な証明書および鍵が含まれるトークンが発行されます。

トークンの登録後、Mozilla Firefox や Thunderbird などのアプリケーションは、トークンを認識して、クライアント認証や S/MIME メールなどのセキュリティー操作に使用するように設定できます。
Enterprise Security Client は、以下の機能を提供します。

- Global Platform 準拠のスマートカードをに対応しています。
- セキュリティートークンを登録して、Red Hat Certificate System のトークン管理システムで認識されるようにします。
- トークンの再登録など、セキュリティートークンを維持します。
- 管理対象トークンの現在のステータスに関する情報を提供します。
- トークンが失われた場合に別のトークンで鍵をアーカイブおよび復元できるように、Certificate System サブシステムによるサーバー側の鍵生成をサポートします。

1.1. RED HAT ENTERPRISE LINUX、シングルサインオン、および認証

ネットワークユーザーは、使用する各種サービスに複数のパスワードを送信する必要があります (電子メール、Web ブラウジ、イントラネット、およびネットワーク上のサーバーなど)。複数のパスワードを維持して、これらの入力を使い分けることは、ユーザーおよび管理者にとって困難です。**シングルサインオン** は、管理者が単一のパスワードストアを作成してユーザーが一度ログインし、単一のパスワードを使用してすべてのネットワークリソースに認証できるようにするための設定です。

Red Hat Enterprise Linux 6 は、ワークステーションへのログインやスクリーンセーバーのロック解除、Mozilla Firefox を使用して暗号化された Web ページへのアクセス、Mozilla Thunderbird を使用した暗号化された電子メールの送信など、複数のリソースのシングルサインオンをサポートします。

シングルサインオンは、ユーザーにとって利便でかつ、サーバーおよびネットワークにさらなる層のセキュリティーを確立できます。シングルサインオンは、セキュアで効果的な認証をオンにします。Red Hat Enterprise Linux は、シングルサインオンを有効にするために使用できる認証メカニズムを 2 つ提供します。

- Kerberos ベースの認証
- Red Hat Certificate System が実装する公開鍵インフラストラクチャーに関連付けられるエンタープライズセキュリティークライアントを使用したスマートカードベースの認証

セキュアなネットワーク環境を確立するための第一歩は、ネットワークへのアクセス権限を持つユーザーにアクセスが制限されるようにすることです。アクセスが許可されると、ユーザーはシステムに対して **認証**できます。つまり、ユーザーはアイデンティティーを検証できるということです。

多くのシステムは、Kerberos を使用して **チケット** と呼ばれる有効期限の短い認証情報のシステムを確立します。この認証情報は、ユーザー要求で生成されるアドホックが生成されます。ユーザーは、ユーザーを特定し、チケットを発行できるシステムを示す username-password ペアの形式で認証情報を表示する必要があります。このチケットは、Web サイトや電子メールなどの他のサービスが繰り返し参照できます。これには、ユーザーが単一の認証プロセスを経由する必要があります。

アイデンティティーを検証する代替的な方法は、証明書を提示することです。証明書とは、提示するエンティティーを特定する電子ドキュメントです。スマートカードベースの認証では、これらの証明書は、スマートカードまたはトークンと呼ばれる小規模なハードウェアデバイスに保存されます。ユーザーがスマートカードを挿入すると、スマートカードは証明書をシステムに提示し、ユーザーを認証できるように識別します。

スマートカードを使用したシングルサインオンには、以下の 3 つの手順があります。

1. ユーザーがスマートカードをカードリーダーに挿入します。これは、Red Hat Enterprise Linux のプラグ可能な認証モジュール (PAM) により検出されます。

2. システムが証明書をユーザーエントリーにマッピングし、スマートカードで提示された証明書をユーザーエントリーに保存されている証明書と比較します。

3. 証明書がキー配布センター (KDC) に対して正常に確認されると、ユーザーはログインを許可されます。

スマートカードベースの認証は、追加の識別メカニズム (証明書) と物理アクセス要件を追加することで、Kerberos によって確立されtるシンプルな認証レイヤーで構築されます。

1.2. RED HAT CERTIFICATE SYSTEM および ENTERPRISE SECURITY クライアント

Red Hat Certificate System は、証明書および鍵の作成、管理、更新、取り消しを行います。証明書システムには、スマートカードを管理するために、キーの生成、証明書リクエストの作成、および証明書受信を行うためのトークン管理システムがあります。

2 つのサブシステム: Token Key Service (TKS) および Token Processing System (TPS) は、トークン関連の操作を処理するために使用されます。Enterprise Security Client は、スマートカードとユーザーがトークン管理システムにアクセスできるようにするインターフェースです。

4 つの証明書システムサブシステムの合計はトークンの管理に関与します。そのうちの 2 つは、トークンの管理 (KS および TPS)、残りの 2 つは公開鍵インフラストラクチャー (CA および DRM) 内の鍵と証明書を管理です。

- Token Processing System (TPS) はスマートカードと対話し、ユーザーやデバイスなどの特定のエンティティーのキーと証明書を生成および保存できるようにします。スマートカード操作は TPS を経由して、証明書を生成する認証局やデータリカバリーマネージャーなどのアクションのために適切なサブシステムに転送され、キーをアーカイブおよび復元します。

- Token Key Service (TKS) は、TPS とスマートカード間の通信に使用される対称鍵を生成または派生します。TKS によって生成された鍵のセットは、カードの一意の ID を基にしているため一意です。鍵はスマートカードでフォーマットされ、スマートカードと TPS との間で通信の暗号化、または認証を行うために使用されます。

- 認証局 (CA) は、スマートカードに保存されているユーザー証明書を作成して破棄します。

- 必要に応じて、Data Recovery Manager (DRM) アーカイブをアーカイブし、スマートカードのキーを復元します。

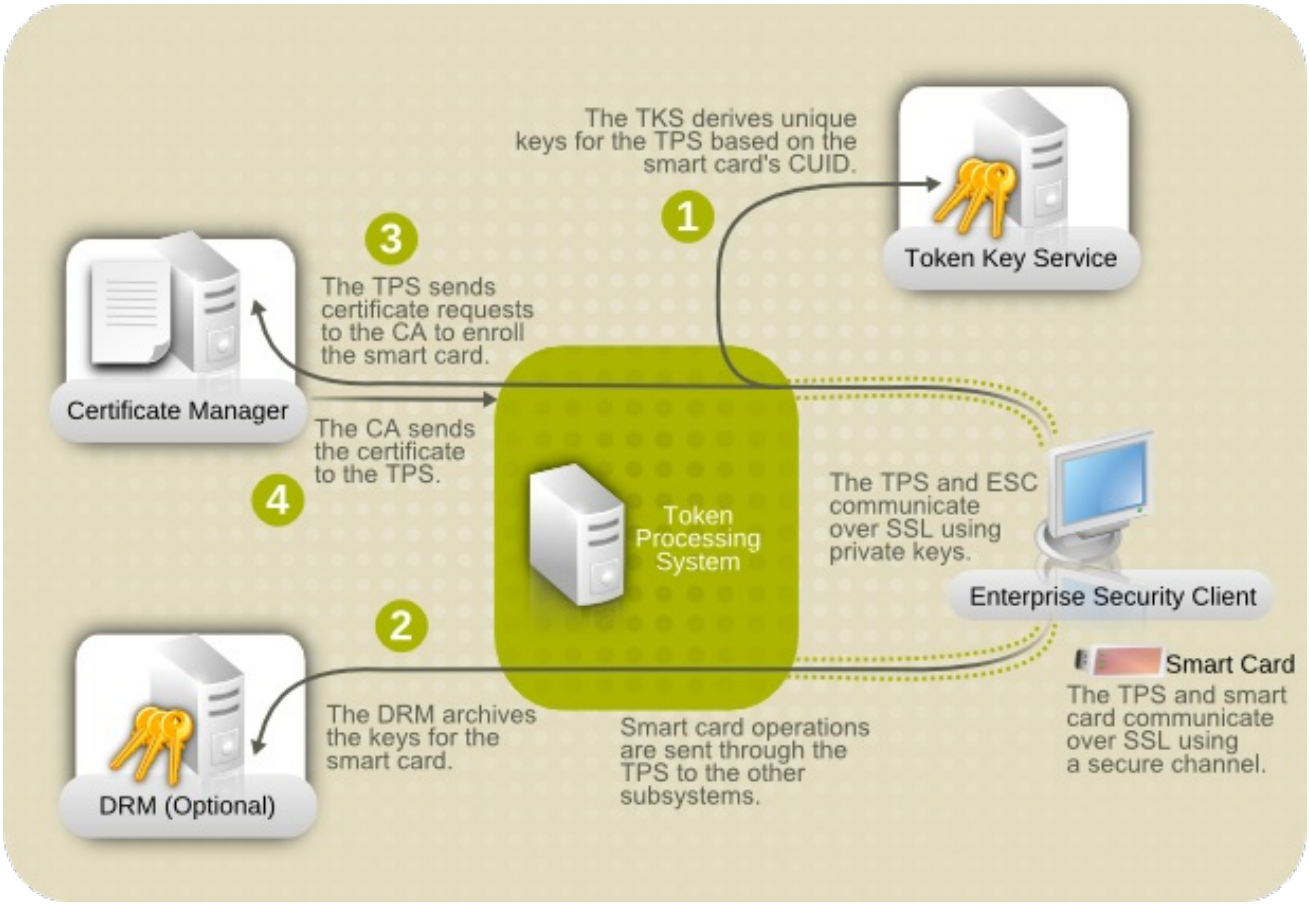


図1.1 証明書システムのスマートカードの管理方法

図1.1「証明書システムのスマートカードの管理方法」 で示すように、TPS は Red Hat Certificate System トークン管理システムの中心となるハブです。トークンは TPS と直接通信します。次に、TPS は TKS と通信して、TAS-token の通信 (1) に使用できる一意の鍵のセットを取得します。スマートカードが登録されると、トークンに新しい秘密鍵が作成されます。キーのアーカイブを設定する場合は、これらのキーを DRM (2) でアーカイブできます。CA は証明書要求 (3) を処理し、トークンに保存する証明書を発行します。TPS は、これらの証明書を Enterprise Security Client (4) に戻します。これらは、トークンに保存されます。

Enterprise Security Client は、TPS が安全な HTTP チャンネル (HTTPS) で各トークンと通信し、証明書システムとともに TPS を介して通信するコンジットです。

トークンを使用するには、Token Processing System がトークンを認識して通信できる必要があります。必要なキーと証明書でトークン作成して、証明書システムにトークンを追加するために、トークンを最初に **登録** する必要があります。Enterprise Security Client は、ユーザーがスマートカードをフォーマットし、管理するためのユーザーインターフェースを提供します。

第2章 PAM (プラグ可能な認証モジュール) の使用

プラグ可能な認証モジュールは、認証およびセキュリティーの共通のフレームワークです。Red Hat Enterprise Linux のシングルサインオンメソッド (Kerberos およびスマートカード) は、基礎となる PAM 設定に依存します。

PAM について理解して使用すると、セキュアで効率的なシングルサインオンソリューションの計画と実装に非常に有利です。

2.1. PAM について

システムにユーザーアクセスを付与するプログラムは、*認証* を使用して相互のアイデンティティーを確認します (つまり、ユーザーがそのユーザーであると確立するためのプログラム)。

これまでは、各プログラムはユーザー認証の方法を使用していました。Red Hat Enterprise Linux では、多くのプログラムが *プラグ可能な認証モジュール* (PAM) と呼ばれる集中型認証メカニズムを使用するように設定されています。

PAM はプラグ可能なモジュラーアーキテクチャーを使用しており、システム管理者はシステム用の認証ポリシーを柔軟に設定することができます。PAM は、開発者および管理者にとって以下のような便利なシステムです。

- PAM は、多様なアプリケーションで使用できる共通の認証スキームを提供します。
- PAM は、システム管理者とアプリケーション開発者の両方に、優れた柔軟性と制御性を提供します。
- PAM は、完全に文書化された単一ライブラリーを提供します。開発者は、独自の認証スキームを作成することなくプログラムの作成ができます。

PAM には、PAM の使用と PAM を他のアプリケーションと統合するためのモジュールの作成に関する詳細が記載されている広範囲なドキュメントセットがあります。PAM を使用する主要なモジュールと設定ファイルのほとんどすべてには、独自の man ページがあります。また、**/usr/share/doc/pam-version#** ディレクトリーには、『System Administrators' Guide』、『Module Writers' Manual』、および『Application Developers' Manual』、および PAM 標準 DCE-RFC 86.0 のコピーが含まれています。

PAM 用のライブラリーは http://www.kernel.org/pub/linux/libs/pam/ で入手できます。これは、Linux-PAM プロジェクトの主なディストリビューション web サイトです。これには、さまざまな PAM モジュールに関する情報、よくある質問、および追加の PAM ドキュメンテーションが含まれています。

2.2. PAM 設定ファイル

/etc/pam.d/ ディレクトリーには、PAM 対応各アプリケーションの PAM 設定ファイルが含まれます。

2.2.1. PAM サービスファイル

PAM 対応のアプリケーションまたは *サービス* にはそれぞれ、**/etc/pam.d/** ディレクトリーにファイルがあります。このディレクトリーの各ファイルは、アクセスを制御するサービスと同じ名前を持ちます。

PAM 対応プログラムは、サービス名を定義し、独自の PAM 設定ファイルを **/etc/pam.d/** ディレクトリーにインストールします。たとえば、**login** プログラムはサービス名を **login** として定義し、**/etc/pam.d/login** PAM 設定ファイルをインストールします。

2.2.2. PAM 設定ファイル形式

各 PAM 設定ファイルには、モジュールとその制御または引数を定義するディレクティブが含まれます。

ディレクティブの構文はすべて簡単なもので、モジュールの目的 (インターフェース) とモジュールの設定を特定します。

```
module_interface     control_flag     module_name module_arguments
```

2.2.2.1. PAM モジュールのインターフェース

4 種類の PAM モジュールインターフェースが利用できます。それぞれは、承認プロセスのさまざまな要素に対応します。

- **auth**: このモジュールインターフェースは、使用を認証します。たとえば、パスワードの有効性を要求し、検証します。このインターフェースのあるモジュールは、グループメンバーシップや Kerberos チケットなどの認証情報を設定することもできます。
- **account**: このモジュールインターフェースは、アクセスが許可されることを確認します。たとえば、ユーザーアカウントの有効期限が切れたか、または特定の時間にユーザーがログインできるかどうかを確認します。
- **password**: このモジュールインターフェースは、ユーザーパスワードの変更に使用されます。
- **session**: このモジュールインターフェースは、ユーザーセッションを設定および管理します。このインターフェースのあるモジュールは、ユーザーのホームディレクトリーをマウントしたり、ユーザーのメールボックスを利用可能にするなど、アクセスを許可するために必要な追加のタスクも実行できます。

注記

個別のモジュールは、いずれかのインターフェースまたはすべてのモジュールインターフェースを提供できます。たとえば、**pam_unix.so** は 4 つのモジュールインターフェースをすべて提供します。

PAM 設定ファイルでは、モジュールインターフェースは以下のように最初のフィールドで定義されます。以下に例を示します。

```
auth required pam_unix.so
```

これにより、PAM が **pam_unix.so** モジュールの **auth** インターフェースを使用するように指示されます。

モジュールインターフェースのディレクティブは、重ねて配置することで**スタック化** が可能なので、複数のモジュールをまとめて 1 つの目的に使用することができます。モジュールの制御フラグが **sufficient** または **requisite** 値を使用している場合、モジュールの一覧表示順序は認証プロセスで重要になります。

スタック化により、管理者はユーザーが認証を行う前に、特定の条件を必須とすることが容易になります。たとえば、**reboot** コマンドは通常、PAM 設定ファイルで説明されているように、いくつかのスタックされたモジュールを使用します。

```
[root@MyServer ~]# cat /etc/pam.d/reboot
```

```
#%PAM-1.0
auth  sufficient pam_rootok.so
auth  required pam_console.so
#auth  include  system-auth
account  required pam_permit.so
```

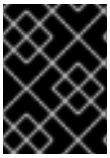
- 最初の行はコメントで、処理されません。
- **auth sufficient pam_rootok.so**: この行は **pam_rootok.so** モジュールを使用して、UID が 0 であることを確認し、現在のユーザーが root かどうかをチェックします。このテストに成功すると、他のモジュールは参照されず、コマンドが実行されます。このテストが失敗すると、次のモジュールが参照されます。
- **auth required pam_console.so**: この行は、**pam_console.so** モジュールを使用してユーザーの認証を試行します。このユーザーがコンソールにすでにログインしている場合は、**pam_console.so** がサービス名 (reboot) と同じ名前を持つ **/etc/security/console.apps/** ディレクトリーにファイルがあるかどうかを確認します。そのようなファイルが存在する場合は、認証が成功し、制御が次のモジュールに渡されます。
- **#auth include system-auth**: この行はコメント化され、処理されません。
- **account required pam_permit.so** - この行は **pam_permit.so** モジュールを使用して、root ユーザーまたはコンソールにログインしているすべてのユーザーによるシステムの再起動を許可します。

2.2.2.2. PAM 制御フラグ

すべての PAM モジュールは、呼び出されると成功または失敗の結果を生成します。コントロールフラグは結果に基づいて PAM に指示します。モジュールは特定の順序でスタックでき、制御フラグは特定モジュールの成功または失敗の重要性を判断し、ユーザーをサービスに対して認証する際の全体的な目標を決定します。

設定にはキーワードのみを使用する単純なフラグがいくつかあります。

- **required**: 認証を継続するには、モジュール結果が成功する必要があります。この時点でテストが失敗すると、そのインターフェースを参照するすべてのモジュールテストの結果が完了するまでユーザーには通知されません。
- **requisite**: 認証を継続するには、モジュール結果が成功する必要があります。ただし、この時点でテストが失敗すると、初回失敗した **required または requisite** モジュールテストを反映したメッセージとともに、すぐにユーザーに通知されます。
- **sufficient**: モジュール結果は、失敗すると無視されます。ただし、モジュールフラグ付きの **sufficient** の結果が成功で、**かつ**、以前のモジュールフラグ付きの **required** が失敗していない場合は、その他の結果は不要で、ユーザーはサービスに認証されます。
- **optional**: モジュール結果は無視されます。その他のモジュールがインターフェイスをを参照しない場合に認証成功に必要となるのは、**optional** としてフラグが付いたモジュールです。
- **include**: 他のコントロールとは異なり、モジュールの結果の処理方法とは関係ありません。このフラグは、指定のパラメーターに一致する設定ファイルのすべての行でプルし、それらをモジュールに引数として追加します。



重要

呼び出される **required** モジュールの順序は重要ではありません。**sufficient** および **requisite** 制御フラグのみにより、順番は重要になります。

設定可能な制御フラグは数多くあります。これらは **attribute=value** のペアで設定されます。属性の全リストは **pam.d** man ページで確認できます。

2.2.2.3. PAM モジュール名

モジュール名は、指定されたモジュールインターフェースを含むプラグ可能なモジュールの名前で PAM を提供します。アプリケーションが適切なバージョンにリンクされているため、ディレクトリー名は省略されます。これは **libpam**、モジュールの正しいバージョンを見つけることができます。

2.2.2.4. PAM モジュール引数

PAM はいくつかのモジュール向けの認証中に *引数* を使って情報をプラグ可能なモジュールに渡します。

たとえば、**pam_userdb.so** モジュールは Berkeley DB ファイルに保存されている情報を使用してユーザーを認証します。Berkeley DB は、多くのアプリケーションに埋め込まれたオープンソースデータベースシステムです。モジュールは **db** 引数を取り、リクエストしたサービスに使用するデータベースを認識できるようにします。以下に例を示します。

```
auth required pam_userdb.so db=/path/to/BerkeleyDB_file
```

無効な引数は通常無視され、PAM モジュールの成功や失敗には影響を及ぼしません。ただし、一部のモジュールは無効な引数で失敗する可能性があります。多くのモジュールはエラーを **/var/log/secure** ファイルに報告します。

2.2.3. PAM 設定ファイルのサンプルについて

例2.1「シンプルな PAM 設定」 は、PAM アプリケーション設定ファイルのサンプルです。

例2.1 シンプルな PAM 設定

```
#%PAM-1.0
auth  required pam_securetty.so
auth  required pam_unix.so nullok
auth  required pam_nologin.so
account  required pam_unix.so
password required pam_cracklib.so retry=3
password required pam_unix.so shadow nullok use_authtok
session  required pam_unix.so
```

- 最初の行は、行頭のハッシュ記号 (**#**) が示すように、コメントになります。
- 2 行目から 4 行目は、ログイン認証用に 3 つのモジュールをスタックしています。

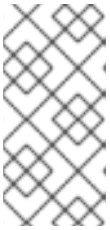
  auth required pam_securetty.so: このモジュールは、ユーザーが root としてログインしようとしている **場合**、そのファイルが**存在すれば**、ユーザーがログインする tty が **/etc/securetty** ファイルに一覧表示されます。

tty がファイルに記載されていない場合は、root でログインしようとすると、**Login incorrect** メッセージとともに失敗します。

auth required pam_unix.so nullok: このモジュールはユーザーにパスワードを要求し、**/etc/passwd** に保存された情報を使用してパスワードをチェックしします。存在する場合は **/etc/shadow**。

引数は **pam_unix.so** モジュールに対し、空のパスワードを許可するように **nullok** に指示します。

- **auth required pam_nologin.so**: これは、認証の最終ステップです。これは、**/etc/nologin** ファイルが存在するかどうかを確認します。ユーザーが存在して root でない場合は、認証に失敗します。



注記

この例では、最初の **auth** モジュールが失敗しても、3 つの **auth** モジュールがすべてチェックされます。これにより、ユーザーは認証に失敗したステージを把握できません。攻撃者のこのような知識により、システムのクラッキング方法がより簡単に推測される可能性があります。

- **account required pam_unix.so**: このモジュールは、必要なアカウントの検証を実行します。たとえば、シャドウパスワードが有効になっていると、**pam_unix.so** モジュールのアカウントインターフェースが、アカウントの有効期限が切れたかどうかや、許可された猶予期間内にユーザーがパスワードを変更していないかどうかを確認します。

- **password required pam_cracklib.so retry=3**: パスワードの有効期限が切れた場合、**pam_cracklib.so** モジュールのパスワードコンポーネントは新しいパスワードを要求します。その後、新たに作成されたパスワードをテストして、辞書ベースのパスワードクラッキングプログラムで簡単に判別できるかどうかを確認します。

  引数 **retry=3** は、テストに 1 回失敗しても、ユーザーは強固なパスワードを作成する機会があと 2 回あることを示しています。

- **password required pam_unix.so shadow nullok use_authtok**: この行は、**pam_unix.so** モジュールの **password** インターフェースを使用して、プログラムがユーザーのパスワードを変更するかどうかを指定します。

  - 引数 **shadow** は、ユーザーのパスワード更新の際にシャドウパスワードを作成するようモジュールに指示します。

  - この引数 **nullok** は、ユーザーが空のパスワード **から**パスワードを変更できるようにするようにモジュールに指示します。それ以外の場合は、null パスワードはアカウントロックとして扱われます。

  - この行の最後の引数 **use_authtok** は、PAM モジュールをスタックする際に順序の重要性を示す優れた引数を提供します。この引数は、ユーザーに新しいパスワードを要求しないようにモジュールに指示します。代わりに、以前のパスワードモジュールで記録されたパスワードを受け入れます。これにより、新しいパスワードはすべて、受け入れられる前に安全なパスワードの **pam_cracklib.so** テストを通過する必要があります。

- **session required pam_unix.so**: 最後の行は、**pam_unix.so** モジュールのセッションインターフェースにセッションを管理するよう指示します。このモジュールは、各セッションの開始と最後で、ユーザー名とサービスタイプを**/var/log/secure** に記録します。このモジュールは、追加機能のために他のセッションモジュールとスタックすることで補足できます。

2.3. PAM モジュールの作成

新しい PAM モジュールは、PAM 対応アプリケーションで使用するためにいつでも作成または追加できます。PAM 対応プログラムは新しいモジュールとメソッドをすぐに使用できます。これは、コンパイルなし、あるいは変更なしで定義します。これにより、開発者およびシステム管理者は、再コンパイルせずに異なるプログラムの認証方法をテストするだけでなく、組み合わせることができるようになります。

モジュール作成に関するドキュメントは、**/usr/share/doc/pam-version#** ディレクトリーに含まれます。

2.4. PAM と管理認証情報のキャッシング

Red Hat Enterprise Linux における数多くのグラフィカル管理ツールは、**pam_timestamp.so** モジュールを使用して、最大 5 分間にわたり上昇した権限をユーザーに提供します。このメカニズムの仕組みを理解することが重要です。これは、**pam_timestamp.so** が有効なときににターミナルから出るユーザーが、コンソールに物理的にアクセスできるユーザーすべてがマシンを変更できる状態のままにするためです。

PAM タイムスタンプスキームでは、グラフィカル管理アプリケーションにより、起動時に root パスワードの入力が求められます。ユーザーが認証されたとき、**pam_timestamp.so** モジュールはタイムスタンプファイルを作成します。デフォルトでは、これは **/var/run/sudo/** ディレクトリーに作成されます。タイムスタンプファイルがすでに存在する場合は、グラフィカル管理プログラムではパスワードの入力が求められません。代わりに、**pam_timestamp.so** モジュールはタイムスタンプファイルを最新の状態にし、ユーザーの不完全な管理アクセスを 5 分追加で保持します。

/var/run/sudo/ユーザー ディレクトリーのファイルを確認して、タイムスタンプファイルの実際の状態を確認できます。デスクトップでは、関連するファイルは **unknown:root** です。これが存在し、タイムスタンプが 5 分未満の場合は、認証情報が有効です。

タイムスタンプファイルが存在すると、パネルの通知スペースに認証アイコンが表示されます。

図2.1 認証アイコン

2.4.1. タイムスタンプファイルの削除

PAM タイムスタンプがアクティブなコンソールを利用する前に、タイムスタンプファイルを破棄することが推奨されます。グラフィカル環境でこれを行うには、パネルの認証アイコンをクリックします。これにより、ダイアログボックスが開きます。**Forget Authorization** ボタンをクリックして、アクティブなタイムスタンプファイルを破棄します。

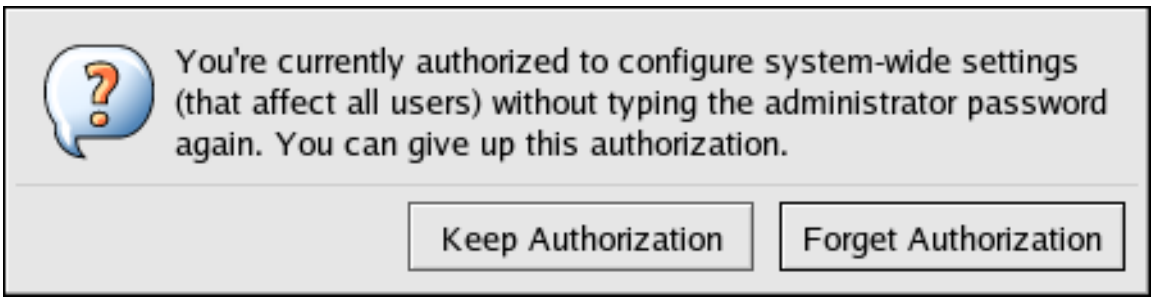


図2.2 認証ダイアログを閉じる

PAM タイムスタンプファイルには、以下の重要な特徴があります。

- **ssh** を使用してシステムにリモートでログインしている場合は、**/sbin/pam_timestamp_check -k root** コマンドを使用してタイムスタンプファイルを破棄します。
- 特権アプリケーションが起動されたものと同じターミナルウィンドウから **/sbin/pam_timestamp_check -k root** コマンドを実行します。
- **pam_timestamp.so** モジュールを最初に起動したログイン済みユーザーは、**/sbin/pam_timestamp_check -k** コマンドを実行するユーザーである必要があります。このコマンドは、root で実行しないでください。
- アイコン上の **Forget Authorization** アクションを使用せずにデスクトップで認証情報を強制終了するには、**/sbin/pam_timestamp_chec** コマンドを使用します。

  ```
  /sbin/pam_timestamp_check -k root </dev/null >/dev/null 2>/dev/null
  ```

  他の方法は、コマンドが実行される pty から認証情報を削除するだけです。

pam_timestamp_check を使用したタイムスタンプファイルの破棄の詳細は、**pam_timestamp_check** man ページを参照してください。

2.4.2. 一般的な pam_timestamp ディレクティブ

pam_timestamp.so モジュールはいくつかのディレクティブを受け付けます。最も一般的なものは以下の 2 つです。

- **timestamp_timeout**: タイムスタンプファイルが有効になる期間 (秒単位) を指定します。デフォルト値は 300 (5 分) です。
- **timestampdir**: タイムスタンプファイルを保存するディレクトリーを指定します。デフォルト値は **/var/run/sudo/** です。

第3章 KERBEROS の使用

ネットワーク内でシステムのセキュリティーと整合性を維持することは重要です。また、ネットワークインフラストラクチャー内のすべてのユーザー、アプリケーション、サービス、およびサーバーが含まれます。これには、ネットワーク上で実行中のすべての内容と、これらのサービスが使用される仕組みを理解する必要があります。このセキュリティーの維持の中核となるのは、これらのアプリケーションおよびサービスへの **アクセス** の維持と、そのアクセスの実施です。

Kerberos は、ユーザーとマシンの両方がネットワークに対して自らを識別し、管理者が設定した領域およびサービスへの定義済みかつ制限されたアクセスを受け取れるようにするメカニズムを提供します。Kerberos はアイデンティティーを確認してエンティティーを**認証**します。また、Kerberos はこの認証データも保護し、外部からアクセス、使用、改ざんされないようにします。

3.1. KERBEROS について

Kerberos は MIT によって作成されたネットワーク認証プロトコルで、対称キー暗号を使用します。[1] ネットワークサービスに対してユーザーを認証します。つまり、パスワードがネットワーク上で送信されることはありません。

そのため、ユーザーが Kerberos を使用してネットワークサービスに対して認証を行う際に、ネットワークトラフィックを監視してパスワードの収集を図っている不正なユーザーを効果的に阻止することができます。

3.1.1. Kerberos の仕組み

従来の多くのネットワークサービスは、パスワードベースの認証スキームを使用しており、ユーザーは特定のネットワークサーバーにアクセスするためのパスワードを提供します。ただし、多くのサービスに対する認証情報の送信は暗号化されません。このようなスキームをセキュアにするには、ネットワークを外部からアクセスできないようにする必要があり、ネットワーク上のすべてのコンピューターおよびユーザーが信頼でき、信頼できるものでなければなりません。

シンプルなパスワードベースの認証では、インターネットに接続されているネットワークが安全であると想定することはできません。ネットワークへのアクセスを取得する攻撃者は、シンプルなパケットアナライザー (**パケットスニッファー**) を使用してユーザー名とパスワードを傍受し、ユーザーアカウントを破ります。そのため、セキュリティーインフラストラクチャー全体の整合性が保たれます。

Kerberos はネットワーク経由で暗号化されていないパスワード送信をなくし、攻撃者がネットワークを傍受する潜在的脅威を取り除きます。

シンプルなパスワード認証で各ユーザーを個別に認証するのではなく、Kerberos は対称暗号化と信頼できるサードパーティー (**キー配布センター** KDC) を使用してユーザーをネットワークサービスのスイートに対して認証します。その KDC とセカンダリー KDC が管理するコンピューターが **レルム** を構成します。

ユーザーが KDC に対して認証を行うと、KDC はそのセッションに特定した認証情報のセット (**チケット**) をユーザーのマシンに送り返します。Kerberos 対応のサービスでは、ユーザーがパスワードを使用して認証する必要はなく、サービスすべてがユーザーのマシン上でこのチケットを探します。

図3.1「Kerberos 認証手順」 にあるように、各ユーザーは **プリンシパル** と呼ばれる一意のアイデンティティーで KDC に識別されます。Kerberos 対応のネットワーク上のユーザーがワークステーションにログインすると、認証サーバーから **ticket-getting ticket** (TGT) の要求の一部としてプリンシパルが KDC に送信されます。この要求は、ログインプログラムによりユーザーに対して透過されるようにしたり、ユーザーがログインした後に **kinit** プログラムを介して手動で送信したりできます。

次に KDC はデータベース内でプリンシパルをチェックします。プリンシパルが見つかると、KDC は TGT を作成し、ユーザーのキーを使用してこれを暗号化し、TGT をそのユーザーに送信します。

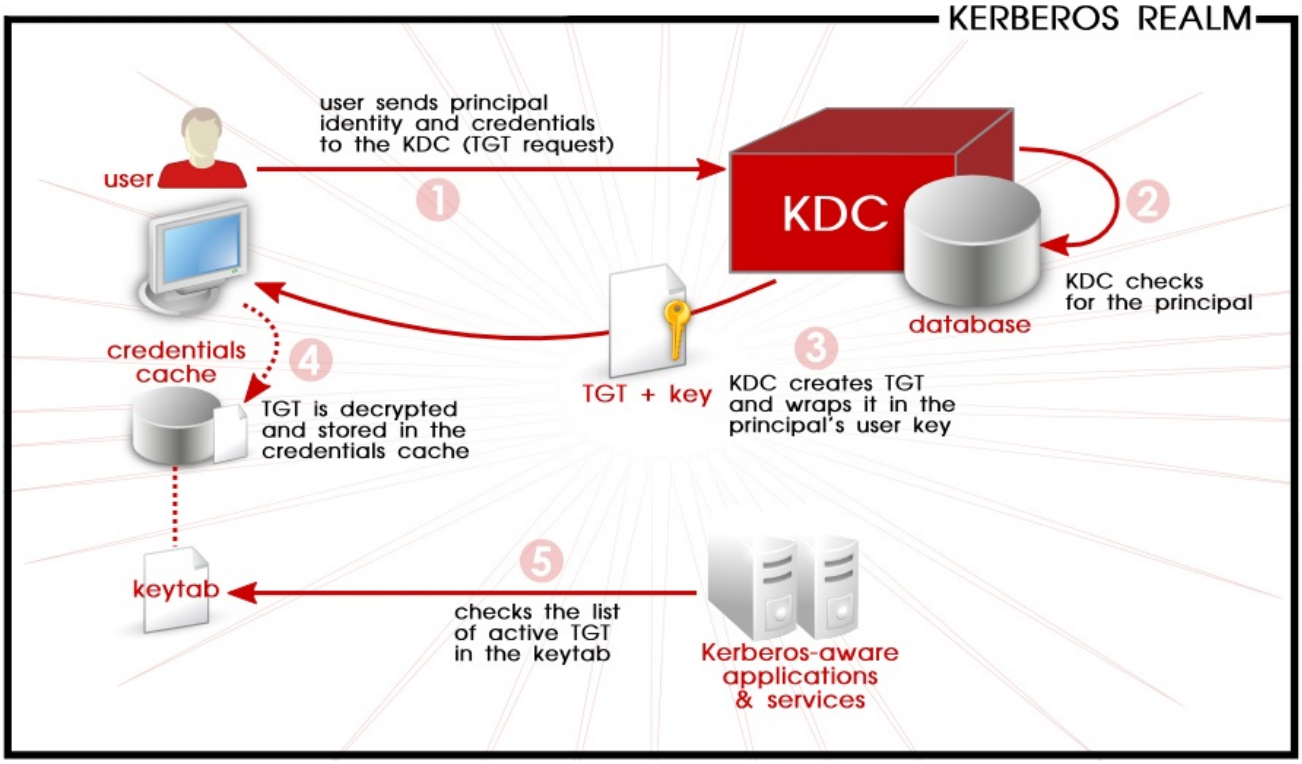


図3.1 Kerberos 認証手順

クライアント上のログインまたは **kinit** プログラムはユーザーの鍵を使用して TGT を復号化します。これは、ユーザーのパスワードから計算します。ユーザーのキーはクライアントマシン上でのみ使用され、ネットワーク上では送信**されません**。KDC が送信するチケット (または認証情報) は、ローカルファイルである **認証情報キャッシュ** に保存され、Kerberos 対応のサービスで確認できます。

認証後、サーバーは、**kinit** ではなく、認識されたプリンシパルとそのキーの暗号化されていないリストをチェックできます。これは **キータブ** に保持されます。

TGT は、一定期間 (通常は 10 から 15 時間) の後に期限切れに設定され、クライアントマシンの認証情報キャッシュに保存されます。セキュリティーの破られた TGT が攻撃者に利用される時間を短くするために、有効期限が設定されています。TGT の発行後、TGT の有効期限が切れるまで、もしくはログアウトして再度ログインするまで、ユーザーはパスワードを再入力する必要はありません。

ユーザーがネットワークサービスにアクセスする必要がある場合、クライアントソフトウェアは TGT を使用して ticket-granting サーバー (TGS) からその特定のサービスの新しいチケットを要求します。サービスチケットはその後、そのサービスに対して透過的にユーザーを認証するために使用されます。

警告

ネットワーク上のユーザーが Kerberos 以外の対応サービスに対してパスワードをプレーンテキストで送信して認証すると、Kerberos システムが危険にさらされる可能性があります。Kerberos 以外の対応サービス (telnet や FTP など) の使用は強く推奨されません。SSH や SSL で保護されたサービスなどの他の暗号化プロトコルは暗号化されていないサービスよりも推奨されますが、これは理想的ではありません。

Kerberos はマシン名を解決でき、正確なタイムスタンプに基づいて発行および期限切れのチケットを発行できる必要があります。そのため、Kerberos が正常に機能するには、十分なクロック同期と作業ドメインネームサービス (DNS) の両方が必要になります。

- ネットワーク上にあるマシン間のクロックの同期の概算は **ntpd** のようなサービスを使用して設定できます。**/usr/share/doc/ntp-version-number/html/index.html** で文書化されています。
- ネットワーク上の DNS エントリーとホストの両方を適切に設定する必要があります。これは、**/usr/share/doc/krb5-server-version-number** の Kerberos ドキュメンテーションで説明されています。

3.1.2. Kerberos 導入における考慮点

Kerberos は一般的かつ重大なセキュリティーの脅威を排除しますが、以下のような理由から実装は容易ではありません。

- **/etc/passwd** または **/etc/shadow** などの標準の UNIX パスワードデータベースから Kerberos パスワードデータベースにユーザーパスワードを移行することは簡単です。このタスクを実行する自動メカニズムはありません。これは、オンラインの Kerberos FAQ の question 2.23 で説明されています。
- Kerberos は、各ユーザーが信頼されていて、信頼できないネットワーク上で信頼できないホストを使用していることを前提としています。その主な目的は、暗号化されていないパスワードがそのネットワーク上で送信されないようにすることです。ただし、認証に使用されるチケットを発行するホスト (KDC) に適切なユーザー以外のユーザーがアクセスすると、Kerberos 認証システム全体が危険にさらされます。
- アプリケーションが Kerberos を使用するには、そのソースを変更して Kerberos ライブラリーに適切な呼び出しを行う必要があります。この方法で変更したアプリケーションは、*Kerberos 対応* または *Kerberized* として考慮されます。アプリケーションによっては、アプリケーションのサイズや設計により、非常に問題になる場合があります。その他の互換性のないアプリケーションでは、サーバーとクライアントが通信する方法に変更を加える必要があります。ここでも、大幅なプログラミングが必要になる場合があります。デフォルトでは Kerberos サポートのないクローズソースアプリケーションは、多くの場合最も問題となります。
- Kerberos は、すべてまたはなしのソリューションです。ネットワーク上で Kerberos を使用している場合は、Kerberos 以外の対応サービスに転送された暗号化されていないパスワードはすべて危険にさらされます。したがって、このネットワークは Kerberos を使用する利点はありません。Kerberos でネットワークの安全を図るには、暗号化されていないパスワードを送信する**すべての**クライアント/サーバーアプリケーションのバージョンで Kerberos 対応のものを使用するか、そのようなクライアント/サーバーアプリケーションをまったく使用しないかのどちらかにする必要があります。

3.1.3. Kerberos に関するその他のリソース

Kerberos は、さまざまな方法でデプロイできる、実装が複雑なサービスとなり得ます。表3.1「外部の Kerberos 資料」 および 表3.2「重要な Kerberos man ページの」 では、Kerberos の使用に関する詳細情報で最重要もしくは最も有益なソースを一覧表示しています。

表3.1 外部の Kerberos 資料

資料	場所
Kerberos V5 Installation Guide (PostScript と HTML の両方)	/usr/share/doc/krb5-server-**version-number**
Kerberos V5 System Administrator's Guide (PostScript と HTML の両方)	/usr/share/doc/krb5-server-**version-number**
Kerberos V5 UNIX User's Guide (PostScript と HTML の両方)	/usr/share/doc/krb5-workstation-**version-number**
Kerberos: The Network Authentication Protocol (MIT のウェブページ)	http://web.mit.edu/kerberos/www/
Kerberos Frequently Asked Questions (FAQ)	http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html
『Designing an Authentication System: a Dialogue in Four Scenes』。原板は Bill Bryant により 1988 年にリリースされ、1997 年に Theodore Ts'o によって改訂されています。本書は、Kerberos スタイルの認証システムの作成を考慮している 2 人の開発者の会話を表しています。この対話形式のディスカッションを参照することで、Kerberos に完全に精通していない人にとって、良い開始地点となります。	http://web.mit.edu/kerberos/www/dialogue.html
ネットワークに Kerberos を使用する方法についての記事。	http://www.ornl.gov/~jar/HowToKerb.html

man ページのファイルは、**man command_name** を実行して開きます。

表3.2 重要な Kerberos man ページの

man ページ	説明
クライアントアプリケーション	
Kerberos	Kerberos システムの概要では、認証情報がどのように機能するかを説明し、Kerberos チケットの取得および破棄に関する推奨事項を提供します。man ページの下部では、関連する man ページが多数参照されています。

man ページ	説明
kinit	このコマンドを使用して ticket-granting ticket を取得、キャッシュする方法が説明されています。
kdestroy	このコマンドを使用して Kerberos 認証情報を破棄する方法が説明されています。
klist	このコマンドを使用して、キャッシュされた Kerberos 認証情報を一覧表示する方法が説明されています。
管理アプリケーション	
kadmin	このコマンドを使用して Kerberos V5 データベースを管理する方法が説明されています。
kdb5_util	このコマンドを使用して Kerberos V5 データベース上で低レベルの管理機能を作成、実行する方法が説明されています。
サーバーアプリケーション	
krb5kdc	Kerberos V5 KDC に利用可能なコマンドラインオプションが説明されています。
kadmind	Kerberos V5 管理サーバー に利用可能なコマンドラインオプションが説明されています。
設定ファイル	
krb5.conf	Kerberos V5 ライブラリー用の設定ファイル内で利用可能な形式とオプションを説明しています。
kdc.conf	Kerberos V5 AS および KDC 用の設定ファイル内で利用可能な形式とオプションを説明しています。

3.2. KERBEROS のインストール

Kerberos パッケージはデフォルトでインストール可能ですが、設定している Kerberos サーバーまたはクライアントに適切なパッケージがインストールされていることを確認します。

Kerberos サーバーのパッケージをインストールするには、以下を実行します。

```
# yum install krb5-server krb5-libs krb5-auth-dialog
```

Kerberos クライアントのパッケージをインストールするには、以下を実行します。

```
# yum install krb5-workstation krb5-libs krb5-auth-dialog
```

Red Hat Enterprise Linux システムが、スマートカードとのシングルサインオンの一部として Kerberos を使用する場合は、必要な PKI/OpenSSL パッケージをインストールします。

```
# yum install krb5-pkinit-openssl
```

3.3. KERBEROS 5 サーバーの設定

Kerberos を設定する際は、マスター KDC を最初にインストールして設定した後に、必要なセカンダリーサーバーをインストールします。

3.3.1. マスター KDC サーバーの設定

1. Kerberos を設定する前に、時刻同期と DNS がすべてのクライアントおよびサーバーマシンで正しく機能していることを確認します。

   Kerberos サーバーとそのクライアント間の時刻同期に特に注意してください。サーバーとクライアント間の時間差が、設定された制限 (デフォルトでは 5 分) よりも大きい場合、Kerberos クライアントはサーバーに認証できません。この時間同期は、攻撃者が古い Kerberos チケットを使用して有効なユーザーとしてマスカレードしないようにするために必要です。

   NTP のドキュメントは **/usr/share/doc/ntp-version-number/html/index.html** と http://www.ntp.org で参照できます。

2. KDC を実行する専用マシンに **krb5-libs krb5-server** および **krb5-workstation** パッケージをインストールします。このマシンは非常に安全である必要があります。可能な場合は、KDC 以外のサービスを実行しないでください。

3. レルム名ととドメイン間のマッピングを反映するように **/etc/krb5.conf** と **/var/kerberos/krb5kdc/kdc.conf** 設定ファイルを編集します。シンプルなレルムは、*EXAMPLE.COM* と *example.com* のインスタンスを正しいドメイン名で置き換えることで構成できます。これは、正しい形式で大文字と小文字の名前を維持することが確実にでき、KDC を *kerberos.example.com* から Kerberos サーバーの名前に変更することで構成できます。通常、レルム名はすべて大文字で、DNS ホスト名およびドメイン名はすべて小文字になります。これらの設定ファイルの man ページには、ファイル形式に関する詳細が記載されています。

4. **kdb5_util** ユーティリティを使用してデータベースを作成します。

   ```
   /usr/sbin/kdb5_util create -s
   ```

   この **create** コマンドは、Kerberos レルムのキーを保存するデータベースを作成します。**-s** 引数は、マスターサーバーキーを保存する *stash* ファイルを作成します。キーの読み取り元となる stash ファイルがない場合、Kerberos サーバー (**krb5kdc**) は起動時に毎回マスターサーバーのパスワード (このパスワードを使って鍵を再生成できる) を要求します。

5. **/var/kerberos/krb5kdc/kadm5.acl** ファイルを編集します。このファイルは、Kerberos データベース **kadmind** への管理アクセス権限およびそのアクセスレベルを決定するために使用します。ほとんどの組織は、1 行で対応できます。

   ```
   */admin@EXAMPLE.COM  *
   ```

   多くのユーザーは、データベース内で単一のプリンシパルで表されます (**joe@EXAMPLE.COM** などの **NULL** または空のインスタンス)。この設定では、**admin** (例: **joe/admin@EXAMPLE.COM**) のインスタンスを持つ 2 番目のプリンシパルを持つユーザー

は、レルムの Kerberos データベースに対する完全な管理制御を強化できます。

kadmin がサーバーで起動した後、ユーザーはレルム内のいずれかのクライアントまたはサーバーで **kadmind** を実行することで、そのサービスにアクセスできます。ただし、**kadm5.acl** ファイルにリストされているユーザーのみが、自身のパスワードを変更することを除いて、データベースを編集できます。

注記

この **kadmin** ユーティリティーはネットワーク経由で **kadmind** サーバーと通信し、Kerberos を使用して認証を処理します。したがって、ネットワーク経由でサーバーに接続してサーバーを管理するには、最初のプリンシパルがすでに存在している必要があります。**kadmin.local** コマンドを使って最初のプリンシパルを作成します。これは、KDC と同じホストで使用するように特別に設計されており、認証に Kerberos を使用しません。

6. KDC ターミナル **kadmin.local** で最初のプリンシパルを作成します。

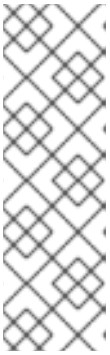
```
/usr/sbin/kadmin.local -q "addprinc username/admin"
```

7. 以下のコマンドを使用して Kerberos を起動します。

```
/sbin/service krb5kdc start
/sbin/service kadmin start
```

8. **kadmin** 内で **addprinc** コマンドを使用してユーザーのプリンシパルを追加します。**kadmin** および **kadmin.local** は、KDC に対するコマンドラインインターフェースです。そのため、**addprinc** などのコマンドの多くは、**kadmin** プログラムの起動後に利用できます。詳細については **kadmin** の man ページを参照してください。

9. KDC がチケットを発行していることを確認します。まず、**kinit** を実行してチケットを取得し、認証情報キャッシュファイルに保存します。次に、**klist** を使用してキャッシュ内の認証情報の一覧を表示し、**kdestroy**を使用して、キャッシュと含まれる認証情報を破棄します。



注記

デフォルトでは、**kinit** が同じシステムログインユーザー名 (Kerberos サーバーではなく) を使用して認証を試みます。ユーザー名が Kerberos データベースのプリンシパルに対応しない場合は、**kinit** がエラーメッセージを発行します。その場合は、コマンドラインの引数として、正しいプリンシパルの名前とともに **kinit** を提供します。

```
kinit principal
```

3.3.2. セカンダリー KDC の設定

あるレルムに複数の KDC がある場合は、1 つの KDC (**マスター KDC**) が書き込み可能なレルムデータベースの書き込み可能なコピーを保持し、**kadmind** を実行します。マスター KDC もレルムの **管理サーバー** です。追加のセカンダリー KDC はデータベースの読み取り専用コピーを維持して、**kpropd** を実行します。

マスタースレーブを伝達するステップでは、マスター KDC がデータベースを一時ダンプファイルにダンプして、そのファイルを各スレーブに送信する必要があります。このファイルは、そのダンプファイルのコンテンツでこれ以前に受信したデータベースの読み取り専用コピーを上書きします。

セカンダリー KDC を設定するには、以下の手順にしたがいます。

1. マスター KDC の **krb5.conf** および **kdc.conf** ファイルをセカンダリー KDC にコピーします。
2. マスター KDC で root シェル **kadmin.local** から開始します。
    1. マスター KDC の **host** サービスの新規エントリーを作成するには、**kadmin.local add_principal** コマンドを使用します。
    2. **kadmin.local ktadd** コマンドを使ってサービス用にランダムの鍵を設定し、その鍵をマスターのデフォルト keytab ファイルに保存します。

注記

この鍵は、**kprop** コマンドがセカンダリーサーバーに認証するために使用されます。インストールするセカンダリー KDC サーバーの数にかかわらず、これは一度だけ実行する必要があります。

```
# kadmin.local -r EXAMPLE.COM
 Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin: add_principal -randkey host/masterkdc.example.com
Principal "host/host/masterkdc.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/masterkdc.example.com
Entry for principal host/masterkdc.example.com with kvno 3, encryption type Triple DES
cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption type ArcFour
with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption type DES with
HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/masterkdc.example.com with kvno 3, encryption type DES cbc
mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: quit
```

3. セカンダリー KDC で root シェルから **kadmin** を起動します。
    1. この **kadmin add_principal** コマンドを使用して、セカンダリー KDC の **ホスト** サービスの新規エントリーを作成します。
    2. **kadmin ktadd** コマンドを使ってサービス用にランダムの鍵を設定し、その鍵をセカンダリー KDC サーバーのデフォルト keytab ファイルに保存します。このキーは、クライアントの認証時に **kpropd** サービスによって使用されます。

```
# kadmin -p jsmith/admin@EXAMPLE.COM -r EXAMPLE.COM
Authenticating as principal jsmith/admin@EXAMPLE.COM with password.
Password for jsmith/admin@EXAMPLE.COM:
kadmin: add_principal -randkey host/slavekdc.example.com
Principal "host/slavekdc.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/slavekdc.example.com@EXAMPLE.COM
Entry for principal host/slavekdc.example.com with kvno 3, encryption type Triple DES
cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption type ArcFour with
```

```
HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption type DES with
HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/slavekdc.example.com with kvno 3, encryption type DES cbc
mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: quit
```

4. セカンダリー KDC はサービスキーを使用して、接続するクライアントをすべて認証できます。すべてのクライアントが新しいレルムデータベースで **kprop** サービスを提供することができるわけではありません。アクセスを制限するため、セカンダリー KDC 上の **kprop** サービスは、プリンシパル名が **/var/kerberos/krb5kdc/kpropd.acl** でリストされているクライアントからのアップデートのみを受け入れます。

   マスター KDC のホストサービス名をこのファイルに追加します。

   ```
   # echo host/masterkdc.example.com@EXAMPLE.COM > /var/kerberos/krb5kdc/kpropd.acl
   ```

5. セカンダリー KDC がデータベースのコピーを取得したら、暗号化に使用したマスターキーも必要になります。KDC データベースのマスターキーがマスター KDC の古いファイル (通常は **/var/kerberos/krb5kdc/.k5.REALM** という名前) に保存されている場合は、利用可能なセキュアな方法を使用してこれをセカンダリー KDC にコピーするか、**kdb5_util create -s** を実行して同じパスワードを指定して、ダミーデータベースを作成して、セカンダリーKDC で同一の古いファイルを作成します。ダミーデータベースは、最初に成功したデータベースの伝播によって上書きされます。

6. セカンダリー KDC のファイアウォールにより、マスター KDC がポート 754 (**krb5_prop**) で TCP を使用して接続し、**kprop** サービスを起動できることを確認します。

7. **kadmin** サービスが **無効** になっていることを再度確認します。

8. マスター KDC のレルムデータベースを、**kprop** コマンドが読み取るデフォルトのデータファイル (**/var/kerberos/krb5kdc/slave_datatrans**) にダンプして、手動でデータベース伝搬テストを実行します。

   ```
   # /usr/sbin/kdb5_util dump /var/kerberos/krb5kdc/slave_datatrans
   ```

9. **kprop** コマンドを使用して、そのコンテンツをセカンダリー KDC に送信します。

   ```
   # kprop slavekdc.example.com
   ```

10. **kinit** を使用して、クライアントシステムが KDC から正常に初回認証情報を取得できることを確認します。クライアントの **/etc/krb5.conf** は、KDC の一覧内のセカンダリー KDC のみを一覧表示する必要があります。

11. レルムデータベースをダンプし、**kprop** コマンドの実行によりデータベースを各セカンダリー KDC に送信するスクリプトを作成します。このスクリプトを定期的に実行するように **cron** サービスを設定します。

3.4. KERBEROS 5 クライアントの設定

Kerberos 5 クライアントの設定に必要となるのは、クライアントパッケージをインストールして各クライアントに有効な **krb5.conf** 設定ファイルを指定することです。クライアントシステムにリモートでログインする方法として、**ssh** と **slogin** が推奨されています。また、Kerberos を使用したバージョンの **rsh** と **rlogin** も、設定を加えることで利用できます。

1. Kerberos クライアントと KDC との間で時刻同期が行われており、DNS が Kerberos クライアントで適切に機能していることを確認してください。

2. **krb5-libs** および **krb5-workstation** パッケージをすべてのクライアントマシンにインストールします。

3. 各クライアントに有効な **/etc/krb5.conf** ファイルを提供します (通常は KDC で使用される **krb5.conf** ファイルと同じファイルになります)。

4. Kerberos を適用した **rsh** と **rlogin** サービスを使用するには、**rsh** パッケージをインストールします。

5. ワークステーションが Kerberos を使用して、**ssh**、**rsh**、または **rlogin** を氏使用して接続するユーザーを認証する前に、Kerberos データベースに独自のホストプリンシパルが必要になります。**sshd**、**kshd**、および **klogind** サーバープログラムはすべて、ホストサービスのプリンシパルのキーへのアクセスが必要になります。

    1. **kadmin** を使用して、KDC 上のワークステーション用のホストプリンシパルを追加します。この場合のインスタンスはワークステーションのホスト名です。**kadmin** の **addprinc** コマンドに **-randkey** オプションを指定してプリンシパルを作成し、それをランダムな鍵に割り当てます。

        ```
        addprinc -randkey host/server.example.com
        ```

    2. **ワークステーション自体**で **kadmin** を実行して、**ktadd** コマンドを使用すると、鍵をワークステーション用に抽出できます。

        ```
        ktadd -k /etc/krb5.keytab host/server.example.com
        ```

6. その他の Kerberos ネットワークサービスを使用するには、krb5-server パッケージをインストールしてサービスを起動します。Kerberos 化したサービスが 表3.3「一般的な Kerberos 化されたサービス」 に一覧表示されます。

表3.3 一般的な Kerberos 化されたサービス

サービス名	使用方法
ssh	クライアントとサーバー両方の設定で **GSSAPIAuthentication** が有効な場合に、OpenSSH は GSS-API を使用してサーバーにユーザーを認証します。クライアントでも **GSSAPIDelegateCredentials** が有効な場合は、ユーザーの認証情報がリモートシステムで利用可能になります。
rsh および rlogin	**klogin**、**eklogin**、**kshell** を有効化します。
Telnet	**krb5-telnet** を有効化します。
FTP	**ftp** のルートでプリンシパルのキーを作成し、展開します。インスタンスを FTP サーバーの完全修飾ホスト名に設定してから、**gssftp** を有効にするようにしてください。

サービス名	使用方法
IMAP	**cyrus-imap** パッケージもインストールされている場合は、この **cyrus-sasl-gssapi** パッケージは Kerberos 5 を使用します。**cyrus-sasl-gssapi** パッケージには、GSS-API 認証をサポートする Cyrus SASL プラグインが含まれます。Cyrus IMAP 機能は、**cyrus** ユーザーが **/etc/krb5.keytab** で適切な鍵を見つけ、プリンシパルのルートが **imap** (**kadmin** で作成したもの) に設定されている限り Kerberos で動作します。 **cyrus-imap** に変わるものは、**dovecot** パッケージにあります。これは、Red Hat Enterprise Linux にも含まれています。このパッケージには IMAP サーバーが含まれていますが、現在のところ GSS-API および Kerberos には対応していません。
CVS	**gserver** は、**cvs** のルートを持つプリンシパルを使用し、それ以外は CVS **pserver** と同じです。

3.5. スマートカード用の KERBEROS クライアントの設定

スマートカードは Kerberos との使用が可能ですが、スマートカード上で X.509 (SSL) ユーザー証明書を認識するための追加設定が必要になります。

1. 他のクライアントパッケージと共に、必要となる PKI/OpenSSL パッケージをインストールします。

```
[root@server ~]# yum install krb5-pkinit-openssl
[root@server ~]# yum install krb5-workstation krb5-libs krb5-auth-dialog
```

2. **/etc/krb5.conf** 設定ファイルを編集して、公開鍵インフラストラクチャー (PKI) のパラメーターを設定の **[realms]** セクションに追加します。***pkinit_anchors*** パラメーターは CA 証明書バンドルファイルの場所を設定します。

```
[realms]
  EXAMPLE.COM = {
    kdc = kdc.example.com.:88
    admin_server = kdc.example.com
    default_domain = example.com
    ...
    pkinit_anchors = FILE:/usr/local/example.com.crt
  }
```

3. スマートカード認証 (**/etc/pam.d/smartcard-auth**) とシステム認証 (**/etc/pam.d/system-auth**) の両方の PAM 設定に PKI モジュール情報を追加します。両方のファイルに追加する行は、以下のとおりです。

```
auth        optional      pam_krb5.so use_first_pass no_subsequent_prompt
preauth_options=X509_user_identity=PKCS11:/usr/lib64/pkcs11/libcoolkeypk11.so
```

3.6. ドメインからレルムへのマッピング

クライアントが特定サーバー上で実行中のサービスにアクセスしようとする際は、サービス名 (**ホスト**) とサーバー名 (**foo.example.com**) は分かっていますが、ネットワーク上には複数のレルムが導入されることが可能なので、クライアントはサービスが存在するレルムの名前を推測する必要があります。

デフォルトでは、レルム名はサーバーのドメイン名をすべて大文字にしたものになります。

```
foo.example.org → EXAMPLE.ORG
foo.example.com → EXAMPLE.COM
foo.hq.example.com → HQ.EXAMPLE.COM
```

設定によっては、これで十分ですが、派生したレルム名は存在しないレルムの名前になります。このような場合は、サーバーの DNS ドメイン名からレルムの名前へのマッピングをクライアントシステムの **krb5.conf** の **domain_realm** セクションで指定する必要があります。以下に例を示します。

```
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

この設定では、2 つのマッピングを指定します。最初のマッピングは、example.com DNS ドメイン内のシステムが **EXAMPLE.COM** レルムに属することを指定します。2 つ目は、名前が example.com のシステムがレルムにあることを指定します。ドメインと特定ホストの違いは、最初のピリオド記号の有無でマークされます。マッピングは DNS に直接保存することもできます。

3.7. レルム間認証の設定

あるレルムのクライアント (通常は、Kerberos を使用して、特定のサーバーシステムで実行しているサーバープロセス) を許可するには、**クロスレルム認証**が必要です。

3.7.1. 基本的な信頼関係の設定

最も単純なケースでは、レルム **A.EXAMPLE.COM** のクライアントが **B.EXAMPLE.COM** レルムのサービスにアクセスするには、両方のレルムが **krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM** という名前のプリンシパルのキーを共有する必要があります。また、両方のキーに同じキーが関連付けられている必要があります。

これを行うには、非常に強固なパスワードまたはパスフレーズを選択し、**kadmin** を使用して両方のレルムにプリンシパルのエントリーを作成します。

```
# kadmin -r A.EXAMPLE.COM
kadmin: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM
Enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
Re-enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
Principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created.
quit

# kadmin -r B.EXAMPLE.COM
kadmin: add_principal krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM
Enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
Re-enter password for principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM":
Principal "krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM" created.
quit
```

get_principal コマンドを使用して、鍵バージョン番号 (**kvno** の値) と暗号化タイプの両方が一致することを確認します。

重要

よくある間違った状況は、管理者がパスワードではなくランダム鍵の割り当てに **add_principal** コマンドの **-randkey** オプションを使用し、最初のレルムのデータベースから新しいエントリーをダンプして 2 番目のレルムにインポートすることです。データベースダンプに含まれる鍵自体がマスターキーを使用して暗号化されるため、レルムデータベースのマスターキーが同一でない限り動作しません。

A.EXAMPLE.COM レルムのクライアントは、**B.EXAMPLE.COM** レルムのサービスに対して認証できるようになりました。別の方法として、**B.EXAMPLE.COM** レルムが **A.EXAMPLE.COM** レルムを**信頼**するようになりました。

デフォルトでは、**クロスレルム信頼は一方向です。B.EXAMPLE.COM** レルムの KDC は、**B.EXAMPLE.COM** レルムのサービスに対して認証を行うために **A.EXAMPLE.COM** からのクライアントを信頼できます。ただし、この信頼は自動的に再処理されないため、**B.EXAMPLE.COM** レルムは **A.EXAMPLE.COM** レルムのサービスに対して認証するために信頼されています。反対方向の信頼を確立するには、両方のレルムが **krbtgt/A.EXAMPLE.COM@B.EXAMPLE.COM** サービスの鍵を共有する必要があります。これは、前述の例とは反対のマッピングのエントリーになります。

3.7.2. 複合信頼関係の設定

直接信頼関係がレルム間の信頼を提供する唯一の方法である場合、複数のレルムを含むネットワークはセットアップが非常に困難になります。幸い、クロスレルムの信頼は推移的です。**A.EXAMPLE.COM** のクライアントが **B.EXAMPLE**.COM のサービスに対して認証可能で、**B.EXAMPLE.COM** のクライアントが **C.EXAMPLE.COM** のサービスに対して認証できる場合は、**C.EXAMPLE.COM が A.EXAMPLE.COM を直接信頼していない場合でも、A.EXAMPLE .COM のクライアントは C.EXAMPLE.COM のサービスに対して認証することもできます。**つまり、相互に信頼する必要がある複数のレルムがあるネットワークでは、セットアップする信頼関係に適切な選択を行うことで、必要な作業量を大幅に削減できる可能性があることを意味します。

クライアントのシステムは、特定のサービスが属するレルムを適切に推測できるように設定する必要があります。また、そのレルム内のサービスの認証情報の取得方法を判断できる必要があります。

最初に、通常、指定されたレルムの特定のサーバーシステムから提供されるサービスのプリンシパル名は以下のようになります。

```
service/server.example.com@EXAMPLE.COM
```

サービス は通常、使用するプロトコルの名前 (LDAP、IMAP、CVS、HTTP など) または **ホスト** です。**server.example.com** は、サービスを実行するシステムの完全修飾ドメイン名です。**example.COM** はレルムの名前です。

サービスが属するレルムを判別するため、クライアントは、ホスト名 (**server.example.com**) または DNS ドメイン名(**.example.com**) をレルム名 (**EXAMPLE.COM**) にマップするために DNS または **/etc/krb5.conf** の **domain_realm** セクションを参照してください。

サービスが属するレルムを決定した後、クライアントは接続する必要のあるレルムのセットを決定する必要があります。そして、サービスに対する認証に使用する認証情報を取得するために、接続する順番を判断する必要があります。

これは 2 つの方法の 1 つで実行できます。最も単純な方法は、共有階層を使用してレルムに名前を付けることです。2 つ目は、**krb5.conf** ファイルで明示的な設定を使用します。

3.7.2.1. 名前の共有階層の設定

明示的な設定を必要としないデフォルトの方法は、共有階層内でレルム名を提供することです。たとえば、**A.EXAMPLE.COM**、**B.EXAMPLE.COM**、および **EXAMPLE.COM** という名前のレルムを想定します。**A.EXAMPLE.COM** レリムのクライアントが **B.EXAMPLE.COM** 内のサービスに対して認証を試みると、デフォルトでは、最初に **EXAMPLE.COM** レルムの認証情報を取得してから、これらの認証情報を使用して **B.EXAMPLE.COM** レルムで使用する認証情報を取得します。

このシナリオのクライアントは、レルム名を DNS 名を処理する可能性があるものとして扱います。これは、自身のレルム名のコンポーネントを繰り返し取り除き、階層内で「上」のレリムの名前を生成します。これは、サーバーレリムの「上」でもある位置に達するまで行われます。この時点で、サービスのレルムに到達するまで、サービスのレルム名のコンポーネントを先頭に追加し始めます。プロセスに関与する各レルムは別の「ホップ」です。

たとえば、**A.EXAMPLE.COM** で認証情報を使用すると、**B.EXAMPLE.COM** のサービスに対して認証を行う際に **A.EXAMPLE.COM → EXAMPLE.COM → B.EXAMPLE.COM** の 3 つのホップがあります。

- **A.EXAMPLE.COM** および **EXAMPLE.COM** が **krbtgt/EXAMPLE.COM@A.EXAMPLE.COM** の鍵を共有
- **EXAMPLE.COM** および **B.EXAMPLE.COM** が **krbtgt/B.EXAMPLE.COM@EXAMPLE.COM** の鍵を共有

もう 1 つの例は、**SITE1.SALES.EXAMPLE.COM** で認証情報を使用して **EVERYWHERE.EXAMPLE.COM** でのサービスに対する認証に、複数のシリーズのホップを利用できます。

```
SITE1.SALES.EXAMPLE.COM →
SALES.EXAMPLE.COM →
EXAMPLE.COM →
EVERYWHERE.EXAMPLE.COM
```

- **SITE1.SALES.EXAMPLE.COM** と **SALES.EXAMPLE.COM** が **krbtgt/SALES.EXAMPLE.COM@SITE1.SALES.EXAMPLE.COM** の鍵を共有
- **SALES.EXAMPLE.COM** と **EXAMPLE.COM** が **krbtgt/EXAMPLE.COM@SALES.EXAMPLE.COM** の鍵を共有
- **EXAMPLE.COM** と **EVERYWHERE.EXAMPLE.COM** が **krbtgt/EVERYWHERE.EXAMPLE.COM@EXAMPLE.COM** の鍵を共有

DEVEL.EXAMPLE.COM や **PROD.EXAMPLE.ORG** など、共通のサフィックスを持たない名前のレリム名間にホップがある場合さえあります。

```
DEVEL.EXAMPLE.COM →
EXAMPLE.COM →
COM →
ORG →
EXAMPLE.ORG →
PROD.EXAMPLE.ORG
```

- **DEVEL.EXAMPLE.COM** と **EXAMPLE.COM** が **krbtgt/EXAMPLE.COM@DEVEL.EXAMPLE.COM** の鍵を共有
- **EXAMPLE.COM** と **COM** が **krbtgt/COM@EXAMPLE.COM** の鍵を共有
- **COM** と **ORG** が **krbtgt/ORG@COM** の鍵を共有
- **ORG** と **EXAMPLE.ORG** が **krbtgt/EXAMPLE.ORG@ORG** を共有

- **EXAMPLE.ORG** および **PROD.EXAMPLE.ORG** が **krbtgt/PROD.EXAMPLE.ORG@EXAMPLE.ORG** を共有

3.7.2.2. krb5.conf のパスの設定

より複雑でも柔軟な方法は、**/etc/krb5.conf** の **capaths** セクションを設定することです。これにより、1 つのレルムに認証情報があるクライアントが、チェーン内でどのレルムが次であるかを検索できます。これはい最終的にサーバーに認証できます。

capaths セクションの形式は比較的簡単です。セクションの各エントリーは、クライアントが存在する可能性があるレルムの後に名前が付けられます。そのサブセクション内では、クライアントが認証情報を取得する必要のある中間レルムのセットが、サービスが置かれるレルムに対応するキーの値として一覧表示されます。中間レルムがない場合は、「.」が使用されます。

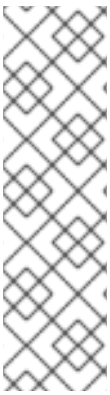
以下に例を示します。

```
[capaths]
A.EXAMPLE.COM = {
B.EXAMPLE.COM = .
C.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = C.EXAMPLE.COM
}
```

A.EXAMPLE.COM レルムのクライアントは、**A.EXAMPLE.COM** KDC から直接 **B.EXAMPLE.COM** のレルム間認証情報を取得できます。

これらのクライアントが **C.EXAMPLE.COM** レルムのサービスに問い合わせる必要がある場合は、最初に **B.EXAMPLE.COM** レルムから必要な認証情報を取得する必要があります (これには **krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM** が必要です)。次に、**これらの**認証情報を使用して **C.EXAMPLE.COM** レルムで使用する認証情報を取得します (**krbtgt/C.EXAMPLE.COM@B.EXAMPLE.COM** を使用)。

これらのクライアントが **D.EXAMPLE.COM** レルムのサービスに問い合わせたい場合は、最初に **B.EXAMPLE.COM** レルムから必要な認証情報を取得し、次に **C.EXAMPLE.COM** レルムから認証情報を取得して **D.EXAMPLE.COM** レルムで使用する認証情報を取得する必要があります。



注記

特に示される capath エントリーがない場合、Kerberos はレルム間の信頼関係が階層を形成していると仮定します。

A.EXAMPLE.COM レルムのクライアントは、**B.EXAMPLE.COM** レルムから直接レルム間の認証情報を取得できます。「.」を指定しないと、クライアントは代わりに階層パスを使用しようとします。この場合は以下のようになります。

```
A.EXAMPLE.COM → EXAMPLE.COM → B.EXAMPLE.COM
```

[1] クライアントとサーバーの両方が共通の鍵を共有しているシステム。これは、ネットワーク通信の暗号化と復号化に使用されます。

第4章 ENTERPRISE SECURITY CLIENT の設定

以下のセクションでは、Enterprise Security Client を使用したトークンの登録、フォーマット、およびパスワードのリセット操作に関する基本的な手順を説明します。

4.1. スマートカードパッケージグループのインストール

esc などのスマートカードの管理に使用するパッケージは、Red Hat Enterprise Linux システムにインストールされているはずです。パッケージがインストールされていないか、または更新する必要がある場合は、**Smart card support** パッケージグループをインストールして、スマートカード関連のパッケージをすべて取り込むことができます。以下に例を示します。

```
yum groupinstall "Smart card support"
```

4.2. スマートカードマネージャーの UI の起動

Enterprise Security Client UI を起動するには、以下の 2 つの側面があります。Enterprise Security Client プロセスを開始して、挿入されたスマートカードまたはトークンを検出できるようサイレントに実行される必要があります。スマートカードが挿入されたり、または手動で開ける場合は、Enterprise Security Client 用のスマートカードマネージャーの UI が自動的に開きます。

コマンドラインから Enterprise Security Client デーモン (**escd**) を起動します。

```
esc
```

このデーモンはスマートカードをサイレンにリッスンし、スマートカードが挿入されるとすぐに GUI を開きます。

Smart Card Manager GUI を手動で開くには、**アプリケーション**、**システムツール**、および**Smart Card Manager** をクリックします。

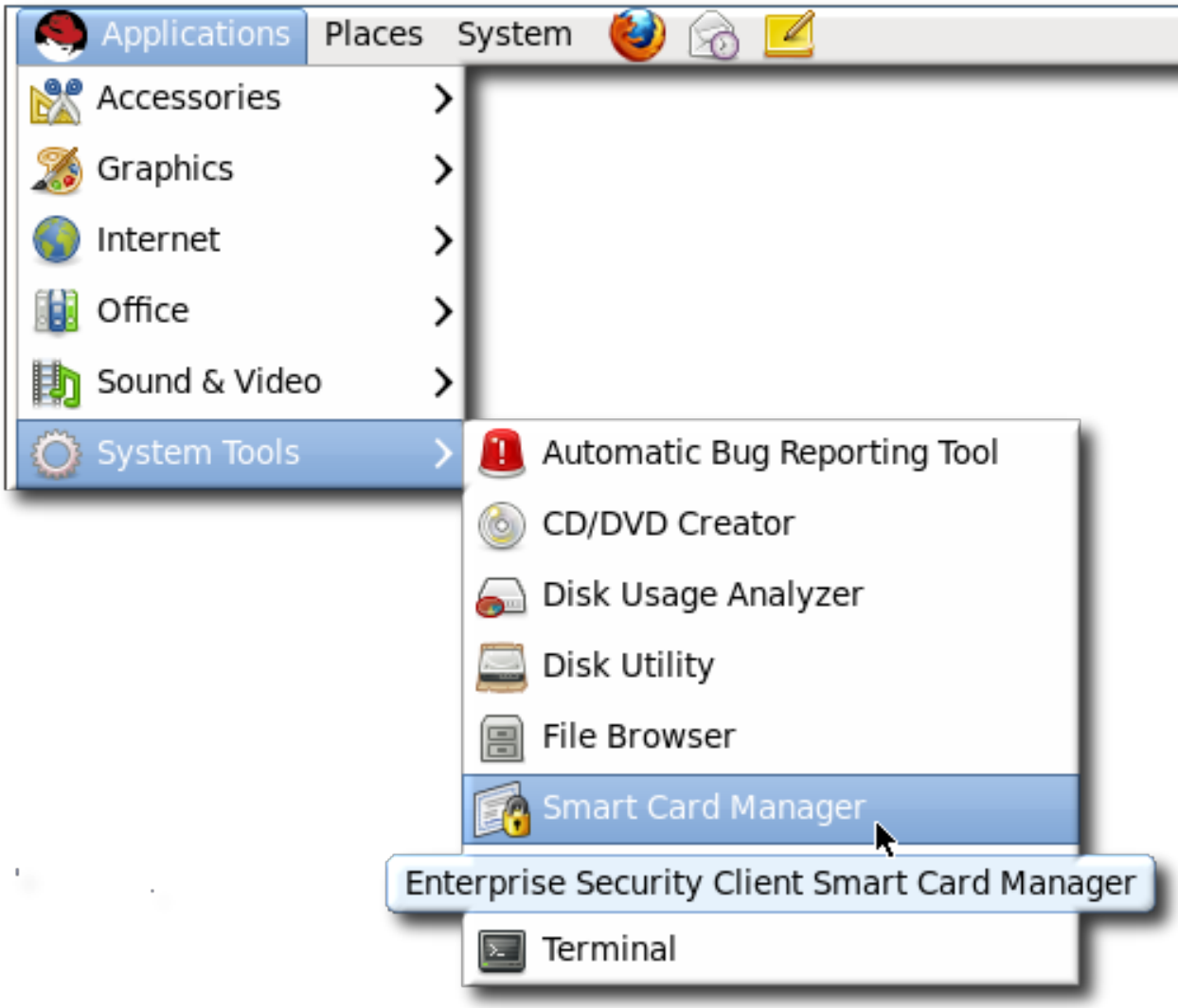


図4.1 メニューの Smart Card Manager アイテムの選択

4.3. ENTERPRISE SECURITY CLIENT 設定の概要

Enterprise Security Client は中間フロントエンドで、ユーザー (およびそのトークン)、トークン処理システム、および認証局との間の接続を提供します。Enterprise Security Client は、以下の 2 つの異なるインターフェースを提供します。

- XUL および JavaScript に基づくローカルインターフェース
- CGI、HTML、および JavaScript に基づくリモートアクセスに使用できる web ホストインターフェース

ローカルサーバーからアクセスされるプライマリー Enterprise Security Client ユーザーインターフェースには、Mozilla XULRunner 技術が含まれています。XULRunner はランタイムパッケージで、ユーザーインターフェース用の機能が充実した XML マークアップ言語である XUL をベースとするスタンドアロンアプリケーションをホストし、アプリケーションの HTML と比較していくつかの利点があります。

- 幅広い UI ウィジェットセット。およびプレゼンテーションにわたる優れた制御。
- クライアントマシンへのローカルマークアップにより、HTML よりも権限レベルが高くなります。
- 便利なプログラム論理スクリプトや XPCOM 技術を利用できるスクリプト言語、JavaScript。

Web ホストインターフェースのすべてのファイルをカスタマイズおよび編集し、Enterprise Security Client の動作や外観を理由に合わせて変更できます。

Token Processing System とともに Enterprise Security Client は、各種ユーザーがさまざまなトークン登録パスを利用できるように、さまざまな **ユーザープロファイル** をサポートします。Enterprise Security Client と TPS は、証明書設定がさまざまなタイプのトークンにカスタム定義できるように、両者とも 異なる**トークンプロファイル** もサポートしています。これらの設定はいずれも TPS で設定されており、『証明書システム管理者のガイド』で説明されています。

4.3.1. Enterprise Security Client ファイルの場所

このリファレンスは、さまざまなクライアントマシンのディレクトリーおよびファイルの場所を示しています。

Red Hat Enterprise Linux 32 ビットでは、Enterprise Security Client はバイナリー RPM によってデフォルトの場所 **/usr/lib/esc-1.1.0/esc** にインストールされます。Red Hat Enterprise Linux 64 ビットシステムでは、インストールディレクトリーは **/usr/lib64/esc-1.1.0/esc** になります。

注記

Enterprise Security Client は特定の XUL 設定ファイルを使用しますが、全体的に、Enterprise Security Client は Red Hat Enterprise Linux で システム XULRunner パッケージを使用します。

表4.1 Enterprise Secuirty Client ファイルおよびディレクトリーの場所

ファイルまたはディレクトリー	目的
application.ini	XULRunner アプリケーション設定ファイル
components/	XPCOM コンポーネント
chrome/	Chrome コンポーネント用のディレクトリーと、Enterprise Security Client XUL および JavaScript の追加アプリケーションファイル。
defaults/	Enterprise Security Client のデフォルト設定。
esc	Enterprise Security Client を起動するスクリプト。

4.3.2. 設定ファイルについて

Enterprise Security Client は、設定ファイルを使用して Mozilla アプリケーションと同様に設定されます。主な設定ファイルは **esc-prefs.js** で、Enterprise Security Client とともにインストールされます。2 つ目は、Enterprise Security Client の初回起動時に作成される Mozilla プロファイルディレクトリーにある **prefs.js** です。

Enterprise Security Client は、サポートされるプラットフォームごとに Mozilla 設定を使用します。Red Hat Enterprise Linux 32 ビットのデフォルト設定ファイルは **/usr/lib/esc-1.1.0/defaults/preferences/esc-prefs.js** にあります。Red Hat Enterprise Linux 64 ビットの場合は、**/usr/lib64/esc-1.1.0/defaults/preferences/esc-prefs.js** にあります。

この **esc-prefs.js** ファイルは、Enterprise Security Client の初回起動時に使用するデフォルト設定を指定します。これには、TPS サブシステムの接続、パスワードのプロンプトの設定Phone Home 情報の設定などを行うパラメーターが含まれます。各設定は **pref** で示され、パラメーターと値は括弧で囲まれます。以下に例を示します。

```
pref(parameter, value);
```

esc-prefs.js ファイルパラメーターが 表4.2「esc-prefs.js Parameters」 に一覧表示されます。デフォルトの **esc-prefs.js** ファイルが 例4.1「デフォルトの esc-prefs.js ファイル」 に表示されます。

表4.2 esc-prefs.js Parameters

パラメーター	説明	注記およびデフォルト
toolkit.defaultChromeURI	XUL Chrome ページへのアクセスに使用するエンタープライズセキュリティークライアントの URL を定義します。	("toolkit.defaultChromeURI", "chrome://esc/content/settings.xul")
esc.tps.message.timeout	TPS に接続するためのタイムアウト期間を秒単位で設定します。	("esc.tps.message.timeout","90");
esc.disable.password.prompt	パスワードプロンプトを有効にします。これは、スマートカードから証明書情報を読み取るのにパスワードが必要であることを意味します。 パスワードプロンプトはデフォルトで無効になっているため、Enterprise Security Client を使用できます。ただし、ある企業がセキュリティー担当者を使用してトークン操作を管理する場合など、セキュリティーコンテキストでは、パスワードプロンプトを有効にして、Enterprise Security Client へのアクセスを制限します。	("esc.disable.password.prompt","yes");

パラメーター	説明	注記およびデフォルト
esc.global.phone.home.url	TPS サーバーへの接続に使用する URL を設定します。 通常、Phone Home 情報はすでにアプレットを介してトークンに設定されます。トークンに Phone Home 情報がない場合 (TPS サーバーと通信する方法がない場合)、Enterprise Security Client はグローバルのデフォルト Phone Home URL をチェックします。 この設定は、明示的に設定されている場合にのみチェックされます。この設定はクライアントでフォーマットされたすべてのトークンに適用されるため、このパラメーターを設定すると、すべてのトークンが同じ TPS をポイントするように強制されます。特定の動作が望ましい場合にのみ、このパラメーターを使用してください。	("esc.global.phone.home.url", "http://server.example.com:7888/cgi-bin/home/index.cgi");
esc.global.alt.nss.db	サーバー上のすべての Enterprise Security Client ユーザーが使用している共通の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を参照します。 Phone Ho me URL この設定は、明示的に設定されている場合にのみチェックされます。これが設定されていない場合、各ユーザーは共有データベースではなく、個別のプロファイルセキュリティーデータベースにのみアクセスします。	prefs("esc.global.alt.nss.db", "C:/Documents and Settings/All Users/shared-db");

例4.1 デフォルトの esc-prefs.js ファイル

このファイルのコメントは、例には含まれません。

```
#pref("toolkit.defaultChromeURI", "chrome://esc/content/settings.xul");
pref("signed.applets.codebase_principal_support",true); for internal use only

pref("capability.principal.codebase.p0.granted", "UniversalXPConnect"); for internal use only
pref("capability.principal.codebase.p0.id", "file://"); for internal use only

pref("esc.tps.message.timeout","90");

#Do we populate CAPI certs on windows?
pref("esc.windows.do.capi","yes");

#Sample Security Officer Enrollment UI
#pref("esc.security.url","http://test.host.com:7888/cgi-bin/so/enroll.cgi");
```

```
#Sample Security Officer Workstation UI
#pref("esc.security.url","https://dhcp-170.sjc.redhat.com:7889/cgi-bin/sow/welcome.cgi");

#Hide the format button or not.
pref("esc.hide.format","no");

#Use this if you absolutely want a global phone home url for all tokens
#Not recommended!
#pref("esc.global.phone.home.url","http:/test.host.com:7888/cgi-bin/home/index.cgi");
```

Enterprise Security Client の起動時に、システム上のユーザーごとに個別の一意のプロファイルディレクトリーを作成します。これらのプロファイルは、Red Hat Enterprise Linux 6 の ~/**.redhat/esc/*alphanumeric_string*.default/prefs.js** に保存されます。

注記

Enterprise Security Client でユーザーの設定値に変更が必要となると、更新された値はデフォルトの JavaScript ファイルではなく、ユーザーのプロファイルエリアに書き込まれます。

表4.3「PREFS.js パラメーター」 は、**prefs.js** ファイルの最も関連するパラメーターを一覧表示します。このファイルの編集は複雑です。この **prefs.js** ファイルは、Enterprise Security Client によって動的に生成および編集されます。このファイルへの手動の変更は、Enterprise Security Client の終了時に上書きされます。

表4.3 PREFS.js パラメーター

パラメーター	説明	注記およびデフォルト
esc.tps.url	TPS への接続に使用する Enterprise Security Client の URL を設定します。これはデフォルトでは設定されません。	
esc.key.*token_ID*.tps.url	TPS との通信に使用するホスト名とポートを設定します。 この Phone Home 情報がファクトリーでカードに書き込まれていない場合は、TPS URL、登録ページの URL、発行者の名前、および Phone Home URL を追加して、カードに手動で追加できます。	("esc.key.**token_ID**.tps.url" = "http://server.example.com:7888/nk_service");
esc.key.**token_ID**.tps.enrollment-ui.url	トークンに証明書を登録する登録ページにアクセスする URL を提供します。 この Phone Home 情報がファクトリーでカードに書き込まれていない場合は、TPS URL、登録ページの URL、発行者の名前、および Phone Home URL を追加して、カードに手動で追加できます。	("esc.key.**token_ID**.tps.enrollment-ui.url" = "http://server.example.com:7888/cgi_bin/esc.cgi?");

パラメーター	説明	注記およびデフォルト
esc.key.**token_ID**.issuer.name	トークンを登録する組織の名前を指定します。	("esc.key.**token_ID**.issuer.name" = "Example Corp");
esc.key.**token_ID**.phone.home.url	TPS の Phone Home 機能にアクセスするために使用する URL を指定します。 トークンが Phone Home 情報を指定しない場合に、グローバル Phone Home パラメーターは、トークン登録で使用するデフォルトを設定します。このパラメーターを特定のトークン ID 番号に設定すると、指定の Phone Home パラメーターがそのトークンにのみ適用されます。	("esc.key.**token_ID**.phone.home.url" = "http://server.example.com:7888/cgi-bin/home/index.cgi?");
esc.security.url	セキュリティー担当者モードに使用する URL を参照します。 これがセキュリティー担当者の登録フォームを参照する場合、Enterprise Security Client はフォームを開き、セキュリティー担当者トークンを登録します。セキュリティー担当者のワークステーション URL を参照する場合は、ワークステーションを開き、セキュリティー担当者の承認で通常のユーザーを登録します。	("esc.security.url","http**s**://server.example.com:7888/**cgi-bin/so/enroll.cgi**");

4.3.3. Enterprise Security Client の XUL ファイルおよび JavaScript ファイルについて

Smart Card Manager は、XUL マークアップおよび JavaScript 機能を **/usr/lib[64]/esc-1.1.0/chrome/content/esc/** に格納します。

プライマリー Enterprise Security Client XUL ファイルは 表4.4「メインの XUL ファイル」 に記載されています。

表4.4 メインの XUL ファイル

ファイル名	目的
settings.xul	**Settings** ページのコードが含まれます。
esc.xul	**登録** ページのコードが含まれます。
config.xul	設定 UI のコードが含まれます。

プライマリー **Smart Card Manager** JavaScript ファイルは次の表に一覧表示されています。

表4.5 メインの JavaScript ファイル

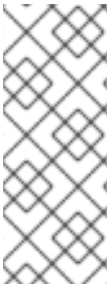
ファイル名	目的
ESC.js	**Smart Card Manager** JavaScript 機能の多くが含まれています。
TRAY.js	トレイアイコン機能が含まれます。
AdvancedInfo.js	**診断**機能のコードが含まれます。
GenericAuth.js	認証プロンプトのコードが含まれています。このプロンプトは、**Smart Card Manager** による動的な処理を必要とする TPS サーバーから設定できます。

4.4. PHONE HOME の設定

Enterprise Security Client の **Phone Home** 機能は、各スマートカード内の情報を、固有の TPS サーバーおよび Smart Card Manager UI ページを示す情報に関連付けます。Enterprise Security Client が新しいスマートカードにアクセスするたびに、TPS インスタンスに接続して、Phone Home 情報を取得できます。

Phone Home はこの情報を取得してキャッシュします。この情報はローカルでキャッシュされているので、フォーマットされたスマートカードが挿入されるたびに TPS サブシステムと通信する必要はありません。

この情報はキーまたはトークンごとに異なる場合があります。つまり、異なる TPS サーバーおよび登録 URL を企業やカスタマーグループごとに設定できます。Phone Home を使用すると、Enterprise Seucirty Client を、正しいサーバーと URL を特定するように手動で設定せずに、発行者や会社単位で異なる TPS サーバーを設定できます。



注記

TPS サブシステムが Phone Home 機能を使用できるようにするには、以下のように TPS 設定ファイルで Phone Home を有効にする必要があります。

```
op.format.userKey.issuerinfo.enable=true
op.format.userKey.issuerinfo.value=http://server.example.com
```

4.4.1. Phone Home プロファイルについて

Enterprise Security Client は Mozilla XULRunner に基づいています。したがって、各ユーザーには、Mozilla Firefox および Thunderbird で使用されるユーザープロファイルと同様のプロファイルがあります。Enterprise Security Client は、設定ファイルにアクセスします。Enterprise Security Client が各トークンの情報をキャッシュすると、情報はユーザーの設定ファイルに保存されます。次回のエンタープライズセキュリティークライアントの起動時に、サーバーを再び接続する代わりに、設定ファイルから情報を取得します。

スマートカードが挿入され、Phone Home を起動すると、Enterprise Security Client は最初にトークンで Phone Home の情報をチェックします。トークンに関する情報がない場合、クライアントは **esc-prefs.js** ファイルで ***esc.global.phone.home.url*** パラメーターを確認します。

トークンに Phone Home 情報が保存されておらず、グローバルな Phone Home パラメーターがない場合、図4.2「Phone Home 情報のプロンプト」 に示されているように、スマートカードが挿入されると、ユーザーは Phone Home URL の入力が求められます。その他の情報は、トークンのフォーマット時に提供され、保存されます。この場合、会社はユーザーに特定の Phone Home URL を提供します。ユーザーが URL を送信すると、フォーマットプロセスにより、残りの情報が Phone Home プロファイルに追加されます。フォーマットプロセスは、ユーザーに変わりません。

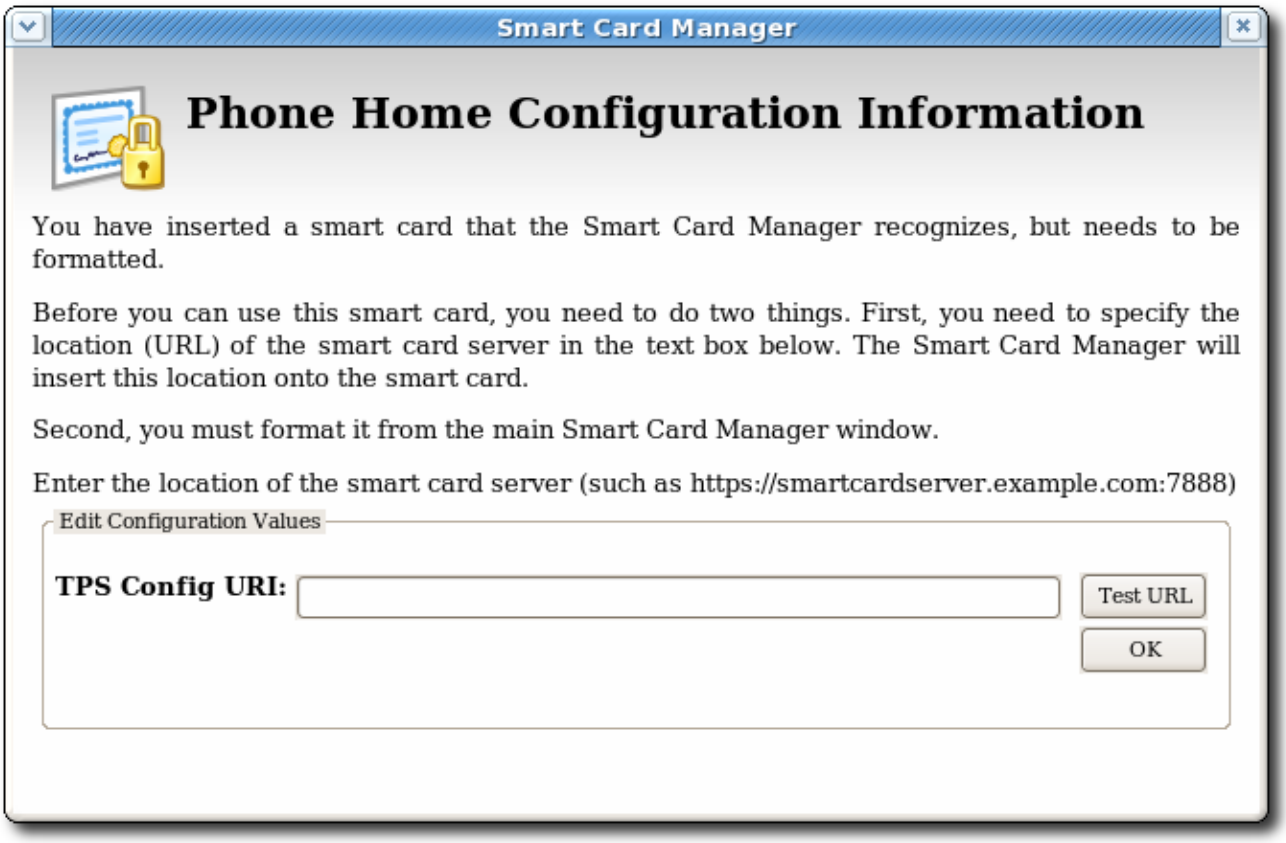


図4.2 Phone Home 情報のプロンプト

4.4.2. グローバル Phone Home 情報の設定

Phone Home は、セキュリティートークンがマシンに挿入されると自動的にトリガーされます。システムは、トークンから Phone Home URL を即座に読み込み、TPS サーバーと通信しようとします。新しいトークンまたは以前にフォーマットされたトークンの場合、Phone Home 情報はカードで利用できない場合があります。

Enterprise Security Client 設定ファイル **esc-prefs.js** には、グローバルな Phone Home URL のデフォルトを設定できるパラメーターがあります。このパラメーターは ***esc.global.phone.home.url*** で、デフォルトではファイルには含まれません。

グローバルの Phone Home URL を定義するには、以下を実行します。

1. 既存の Enterprise Security Client のユーザープロファイルディレクトリーを削除します。プロファイルディレクトリーは、スマートカードが挿入されると自動的に作成されます。デフォルトでは、プロファイルディレクトリーは ~/**.redhat/esc** になります。

2. **esc-prefs.js** ファイルを開きます。

   Red Hat Enterprise Linux 6 では、プロファイルディレクトリーは **/usr/lib/esc-1.1.0/defaults/preferences** になります。64 ビットシステムでは、これは **/usr/lib64/esc-1.1.0/defaults/preferences** になります。

3. グローバルの Phone Home パラメーター行を **esc-prefs.js** ファイルに追加します。以下に例を示します。

   ```
   pref("esc.global.phone.home.url","http://server.example.com:7888/cgi-bin/home/index.cgi");
   ```

   URL は DNS およびネットワーク設定に応じて、マシン名、完全修飾ドメイン名、または IPv4 または IPv6 アドレスを参照できます。

4.4.3. Phone Home 情報の手動によるトークンへの追加

Phone Home 情報は、以下の 2 つの方法の 1 つでトークンに手動で配置できます。

- 推奨される方法は、情報がファクトリーでトークンに書き込まれることです。トークンが製造元から順序付けられると、会社は、送信時にトークンを設定する方法の詳細を提供します。
- トークンが空白である場合、企業の IT 部門は、トークンの小規模なグループのフォーマット時に情報を提供できます。

以下の情報は、~/**.redhat/esc/*alphanumeric_string*.default/prefs.js** ファイル内の各スマートカードの Phone Home 機能によって使用されます。

- TPS サーバーおよびポート以下に例を示します。

  ```
  "esc.key.token_ID.tps.url" = "http://server.example.com:7888/nk_service"
  ```

- TPS 登録インターフェース URL。以下に例を示します。

  ```
  "esc.key.token_ID.tps.enrollment-ui.url" = "http://server.example.com:7888/cgi_bin/esc.cgi?"
  ```

- 発行先の会社名または ID。以下に例を示します。

  ```
  "esc.key.token_ID.issuer.name" = "Example Corp"
  ```

- Phone Home URL。以下に例を示します。

  ```
  "esc.key.token_ID.phone.home.url" = "http://server.example.com:7888/cgi-bin/home/index.cgi?"
  ```

- オプションで、登録したスマートカードが挿入されたときにアクセスするデフォルトのブラウザー URL。

  ```
  "esc.key.token_ID.EnrolledTokenBrowserURL" = "http://www.test.example.com"
  ```

prefs.js ファイルに使用されているパラメーターの詳細は、表4.3「PREFS.js パラメーター」 に記載されています。

注記

これらのパラメーターの URL は DNS およびネットワーク設定に応じて、マシン名、完全修飾ドメイン名、または IPv4 または IPv6 アドレスを参照できます。

4.4.4. TPS を設定し、Phone Home を使用する

Phone Home を使用するように TPS が正しく設定されている場合に限り、Phone Home 機能と、それに使用される各種情報が利用できます。TPS が Phone Home に対して設定されていない場合、この機能は無視されます。Phone Home は **/var/lib/pki-tps/cgi-bin/home** ディレクトリー内の **index.cgi** で設定されています。これにより Phone Home 情報が XML に出力されます。

例4.2「TPS Phone Home 設定ファイル」 は、Phone Home 機能の設定を行う TPS サブシステムによって使用される XML ファイルのサンプルを示しています。

例4.2 TPS Phone Home 設定ファイル

```
<ServiceInfo><IssuerName>Example Corp</IssuerName>
   <Services>
      <Operation>http://server.example.com:7888/nk_service ## TPS server URL
      </Operation>
      <UI>http://server.example.com:7888/cgi_bin/esc.cgi   ## Optional
Enrollment UI
      </UI>
      <EnrolledTokenBrowserURL>http://www.test.url.com   ## Optional
enrolled token url
      </EnrolledTokenBrowserURL>
   </Services>
</ServiceInfo>
```

TPS 設定 URI は、TPS サーバーの URL で、残りのホームディレクトリー情報を Enterprise Security Client に返します。この URL の例は **http://server.example.com:7888/cgi-bin/home/index.cgi** です。この URL はマシン名、完全修飾ドメイン名、IPv4 または IPv6 アドレスを参照できます。TPS 設定 URI にアクセスすると、TPS サーバーは、すべてのホームディレクトリー情報を Enterprise Security Client に返すように求められます。

Smart Card サーバーの URL をテストするには、**TPS Config URI** フィールドにアドレスを入力し、**Test URL** をクリックします。

サーバーが正常に通信されると、メッセージボックスに success が表示されます。テスト接続に失敗すると、エラーダイアログが表示されます。

4.5. セキュリティー担当者モードの使用

エンタープライズセキュリティークライアントは TPS サブシステムとともに、特別な **セキュリティー担当者** モードをサポートします。このモードでは、セキュリティー担当者であるスーパーバイザーが可能になり、特定の組織内の通常ユーザーの登録に直面する機能も受けられます。

セキュリティー担当者モードでは、その他のユーザーのスマートカードをフェイスツーフェイスかつ非常に安全な操作で管理できるセキュリティー担当者の監査下で個人を登録できます。セキュリティー担当者モードは、追加のセキュリティー機能とともに、いくつかの通常のユーザー操作と重複します。

- 組織内の個人を検索する機能。
- 個人関する写真やその他の関連情報を表示するインターフェース。
- 承認された個人の登録機能。
- ユーザーのカードのフォーマットまたはリセット
- セキュリティー担当者のカードのフォーマットまたはリセット
- プライマリーカードを誤って配置したユーザーへの一時カードの登録。
- カードでの TPS サーバー情報の保存この Phone Home 情報は、特定の TPS サーバーインストールに問い合わせるために、Enterprise Security Client によって使用されます。

セキュリティー担当者モードでの操作は、以下の 2 つのエリアに分けられます。

- セキュリティー担当者の作成および管理。
- セキュリティー担当者による通常ユーザーの管理。

セキュリティー担当者モードを有効にすると、Enterprise Security Client はサーバーによって提供される外部ユーザーインターフェースを使用します。このインターフェースは、Enterprise Security Client が通常使用するローカルの XUL コードの代わりにスマートカード操作を制御します。

外部インターフェースは、セキュリティー担当者モードが無効になるまで制御を維持します。

注記

SSL でセキュリティー担当者クライアントを実行すると、TPS が SSL で実行するように設定され、Enterprise Security Client を TPS の SSL エージェントポートに参照することことができるためおすすめです。

4.5.1. セキュリティー担当者モードの有効化

セキュリティー担当者モードが、TPS と Enterprise Security Client の **esc-prefs.js** ファイルの両方で設定される必要がある 2 つの領域があります。

TPS:

1. セキュリティー担当者のユーザーエントリーを TUS オフィスグループのメンバーとして TPS データベースに追加します。このグループは、TPS LDAP データベースにデフォルトで作成され、すべてのセキュリティー担当者ユーザーエントリーの予想される場所です。

注記

Red Hat Directory Server コンソールを使用して、LDAP データベースでユーザーエントリーを追加およびコピーするほうが簡単です。Directory Server Console の使用は、section 3.1.2, Creating Directory Entries の 『Red Hat Directory Server Administrators Guide』 を参照してください。

TPS には 2 つのサブツリーが関連付けられており、それぞれが別のデータベースと関連付けられています。(一般的に、両方のデータベースが同じサーバーに配置することはできますが、必須ではありません。)

- **認証データベース** 内の最初のサフィックスは外部ユーザー用です。TPS はユーザーの認証情報をディレクトリーに対してチェックし、スマートカードの登録を試行するユーザーを認証します。これには、**dc=server,dc=example,dc=com** のような識別名 (DN) があります。

- その他のデータベースは、TPS エージェント、管理者、セキュリティー担当者など、内部 TPS インスタンスエントリーに使用されます。このサブツリーは、TPS の **内部データベース** 内にあります。これには**トークンデータベース**が含まれます。このサブツリーには、**dc=server.example.com-pki-tps** のような TPS サーバーに基づく DN があります。TUS Officer グループエントリーは **dc=server.example.com-pki-tps** 接尾辞の下にあります。

LDAP ディレクトリーとサフィックスは、セキュリティー担当者の auth インスタンスの ***authId*** と ***baseDN*** パラメーターの TPS **CS.cfg** ファイルのトークンプロファイルで定義されます。以下に例を示します。

```
auth.instance.1.authId=ldap2
auth.instance.1.baseDN=dc=sec officers,dc=server.example.com-pki-tps
```

セキュリティー担当者のエントリーは、TUS Officers グループエントリーの子エントリーである必要があります。これは、グループエントリーがメインエントリーで、ユーザーエントリーがディレクトリーツリーのすぐ下にあることを意味します。

TUS Officers グループエントリーは、**cn=TUS Officers,ou=Groups,dc=server.example.com-pki-tps** です。

たとえば、**ldapmodify** を使用してセキュリティー担当者エントリーを追加するには、以下を実行します。

```
/usr/lib/mozldap/ldapmodify -a -D "cn=Directory Manager" -w secret -p 389 -h
server.example.com

dn: uid=jsmith,cn=TUS Officers,ou=Groups,dc=server.example.com-pki-tps
objectclass: inetorgperson
objectclass: organizationalPerson
objectclass: person
objectclass: top
sn: smith
uid: jsmith
cn: John Smith
mail: jsmith@example.com
userPassword: secret
```

Enter キーを 2 回押してエントリーを送信するか、**Ctrl+D** を使用します。

その後、Enterprise Security Client を設定します。

1. まず、CA 証明書チェーンを信頼します。

注記

この手順は、証明書がまだ Enterprise Security Client データベースで信頼されていない場合にのみ必要になります。

Enterprise Security Client を、必要な証明書が含まれるデータベースにポイントする場合は、その **esc-prefs.js** ファイルの ***esc.global.alt.nss.db*** で別のデータベースを指定します。

   1. 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/
```

   2. **Retrieval** タブをクリックして、CA 証明書チェーンをダウンロードします。
   3. Enterprise Security Client を開きます。

```
esc
```

   4. **証明書を表示** ボタンをクリックします。
   5. **認証** タブをクリックします。
   6. **インポート** ボタンをクリックして、CA 証明書チェーンをインポートします。
   7. CA 証明書チェーンの信頼設定を行います。

2. 次に、セキュリティー担当者のトークンをフォーマットし、登録します。このトークンは、セキュリティー担当者のSmart Card Manager UI にアクセスするために使用されます。

   1. 空のトークンを挿入します。
   2. Phone Home 情報のプロンプトが開いたら、セキュリティー担当者の URL を入力します。

```
/var/lib/pki-tps/cgi-bin/so/index.cgi
```

   3. **Format** ボタンをクリックして、セキュリティー担当者のトークンをフォーマットします。
   4. インターフェースを閉じ、Enterprise Security Client を停止します。
   5. **esc-prefs.js** ファイルに 2 つのパラメーターを追加します。最初に、**esc.disable.password.prompt** で、セキュリティー担当者モードを設定します。2 つ目は **esc.security.url** は、セキュリティー担当者の登録ページを参照します。**esc.security.url** パラメーターが存在するだけで、次回の開いた時に、Enterprise Security Client がセキュリティー担当者モードで開くように指示します。

```
pref("esc.disable.password.prompt","no");
pref("esc.security.url","https://server.example.com:7888/cgi-bin/so/enroll.cgi");
```

   6. Enterprise Security Client を再び起動し、UI を開きます。

```
esc
```

7. 新しいセキュリティー担当者のトークンを登録するために、Enterprise Security Client がセキュリティー担当者の登録フォームに接続するよう設定されます。「新しいセキュリティー担当者の登録」 の説明に従ってトークンを登録します。

8. インターフェースを閉じ、Enterprise Security Client を停止します。

9. **esc-prefs.js** ファイルを再度編集します。今回では、**esc.security.url** パラメーターを変更してセキュリティー担当者のワークステーションページを参照します。

```
pref("esc.security.url","https://server.example.com:7889/cgi-bin/sow/welcome.cgi");
```

10. Enterprise Security Client を再起動します。UI はセキュリティー担当者ワークステーションを参照し、セキュリティー担当者が通常のユーザーにトークンを登録できるようになりました。

セキュリティー担当者モードを無効にするには、Smart Card Manager GUI を閉じ、**escd** プロセスを停止して、**esc-prefs.js** ファイル内の **esc.disable.password.prompt** と **esc.security.url** の行をコメントアウトします。**esc** プロセスが再起動すると、通常モードで開始します。

4.5.2. 新しいセキュリティー担当者の登録

セキュリティー担当者は、通常の登録の場合や、セキュリティー担当者が管理する登録に使用されるインターフェースではなく、個別の一意のインターフェースを使用して設定されます。

1. **esc** プロセスが実行していることを確認します。

```
esc
```

esc-pref.js ファイル (「セキュリティー担当者モードの有効化」) でセキュリティー担当者モードが有効になっていると、セキュリティー担当者の登録ページが開きます。

2. **セキュリティー担当者登録** ウィンドウで、新しいセキュリティー担当者の LDAP ユーザー名とパスワードと、セキュリティー担当者のスマートカードで使用するパスワードを入力します。

注記

SSHA ハッシュを使用してパスワードを保存する場合は、パスワードの感嘆符 (!) およびドル記号 ($) 文字が、Windows XP および Vista システムの Enterprise Security Client に正常にバインドされるようにエスケープする必要があります。

- ドル記号 ($) 文字の場合、**パスワードの作成時に** ドル記号をエスケープします。

  ```
  \$
  ```

  次に、Enterprise Security Client にログインするときにドル記号 ($) のみを入力します。

- 感嘆符 (!) 記号では、パスワードの作成時 **および** パスワードの入力時に Enterprise Security Client にログを記録するために文字をエスケープします。

  ```
  \!
  ```

3. **Enroll My Smartcard** をクリックします。

これにより、通常のユーザーがシステム内で登録および管理できるように、セキュリティー担当者が Enterprise Security Client セキュリティー担当者にアクセスするために必要な証明書が含まれるスマートカードが作成されます。

4.5.3. セキュリティー担当者を使用したユーザーの管理

セキュリティー担当者の Station ページでは、新規または一時カードの登録、カードのフォーマット、Phone Home URL の設定などの操作で通常のユーザーを管理します。

4.5.3.1. 新規ユーザーの登録

セキュリティー担当者モードでユーザーのスマートカードを登録するのと、「スマートカードの自動登録」 と 「スマートカードの登録」 でプロセスを登録することには、大きな違いが 1 つあります。すべてのプロセスではユーザーのアイデンティティーを確認するために LDAP データベースにログインが必要です。しかし、セキュリティー担当者モードには、ユーザーが提示する認証情報をデータベース内の一部の情報と比較するための追加のステップがあります (写真など)。

1. **esc** プロセスが実行していることを確認します。必要に応じてプロセスを開始します。

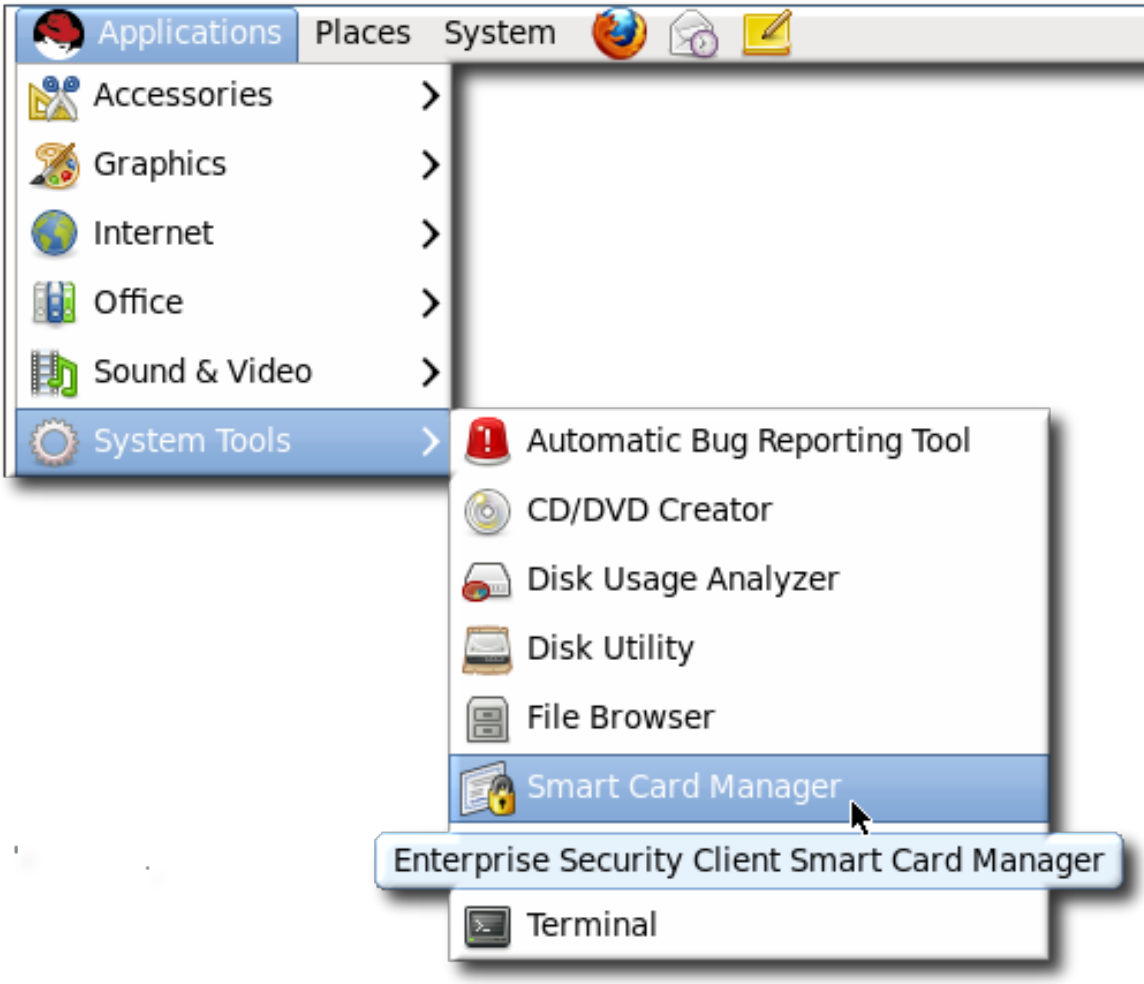
   ```
   esc
   ```

   また、「セキュリティー担当者モードの有効化」 の説明に従って、セキュリティー担当者モードが有効になっていることを確認します。

2. スマートカードマネージャー UI を開きます。

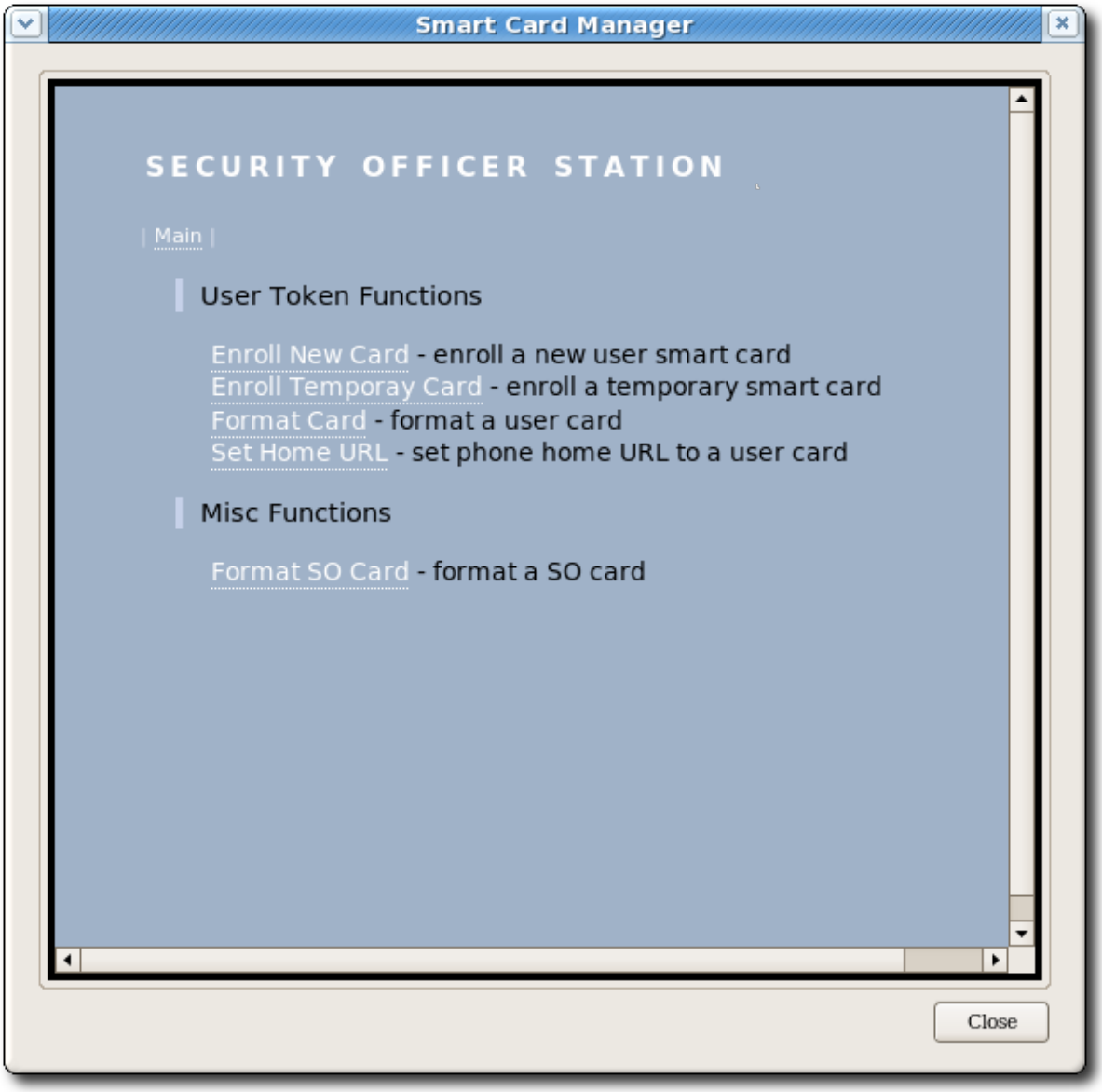
注記

コンピューターに接続したセキュリティー担当者カードが有効で登録されていることを確認します。以下のページにアクセスするには、セキュリティー担当者の認証情報が必要です。



3. **Continue** をクリックすると、セキュリティー担当者の Station ページが表示されます。クライアントはセキュリティー担当者のカード用のパスワード (SSL クライアント認証で必要) を要求します。または、ドロップダウンメニューからセキュリティー担当者の署名証明書を選択するよう要求します。

4. **新規カードの登録** リンクをクリックして、**セキュリティーオフィスの選択**ページを表示します。



5. 新しいスマートカードを受信するユーザーの LDAP 名を入力します。
6. **Continue** をクリックします。そのユーザーが存在する場合は、**Security Officer Confirm User** ページが開きます。
7. Smart Card Manager UI で返される情報と、存在するユーザーまたは認証情報を比較します。
8. すべての詳細が正しい場合は **Continue** をクリックし、**Security Officer Enroll User** ページを表示します。このページでは、担当者に対し、新しいスマートカードをコンピューターに挿入するように促します。
9. スマートカードが適切に認識されると、このカードの新しいパスワードを入力し、**Start Enrollment** をクリックします。

登録に成功すると、スマートカードが作成されたセキュアなネットワークおよびサービスにアクセスするためにユーザーが使用できるスマートカードが生成されます。

4.5.3.2. その他のセキュリティー担当者タスクの実行

セキュリティー担当者が通常ユーザーに対して実行できるその他の操作 (一時トークンの発行、トークンの再登録、または Phone Home URL の設定) は、セキュリティー担当者 UI を開いた後に 4 章*Enterprise security Client の設定* で説明されているとおりに行われます。

1. **esc** プロセスが実行していることを確認します。必要に応じてプロセスを開始します。

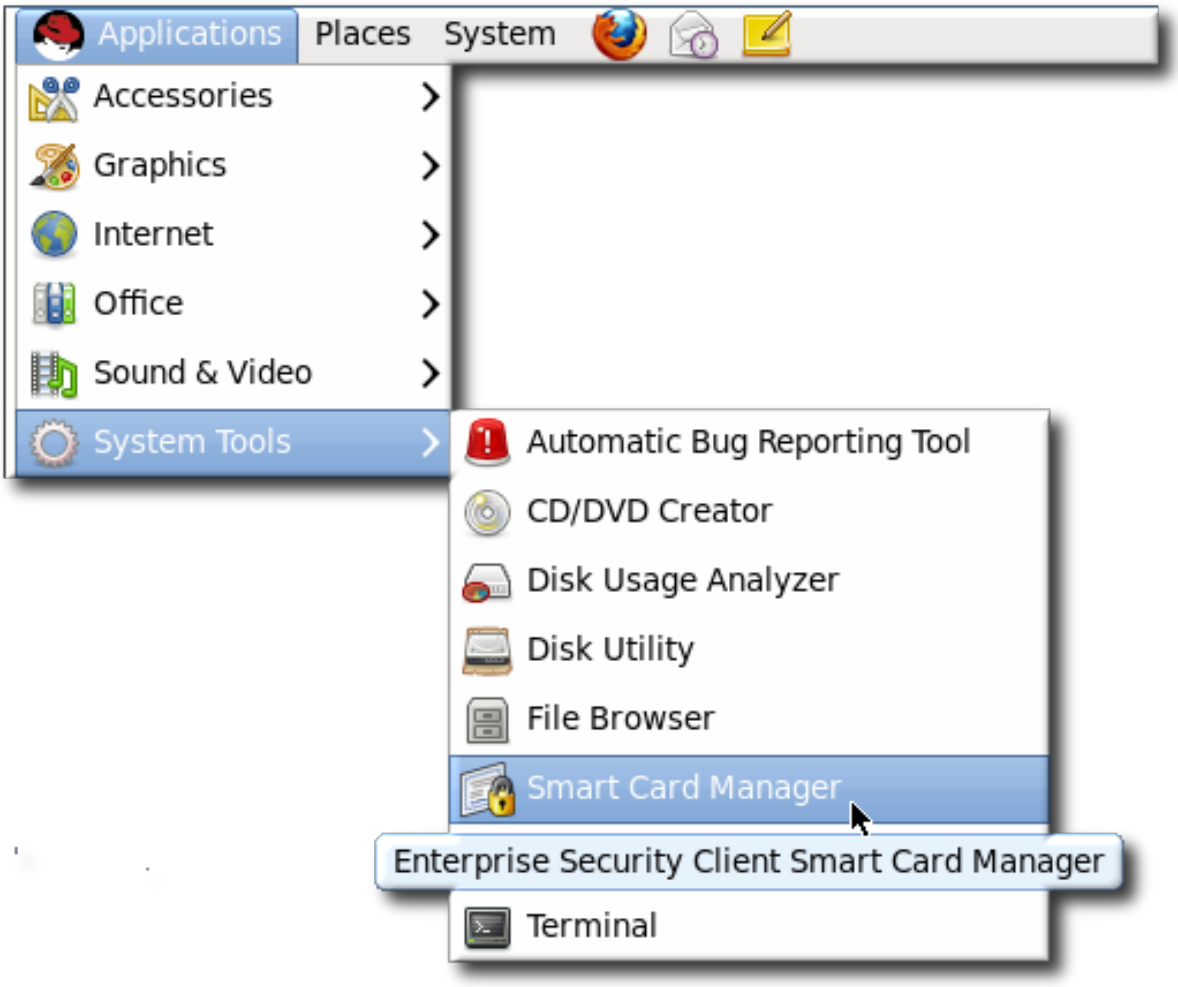
```
esc
```

また、「セキュリティー担当者モードの有効化」 の説明に従って、セキュリティー担当者モードが有効になっていることを確認します。

2. スマートカードマネージャー UI を開きます。

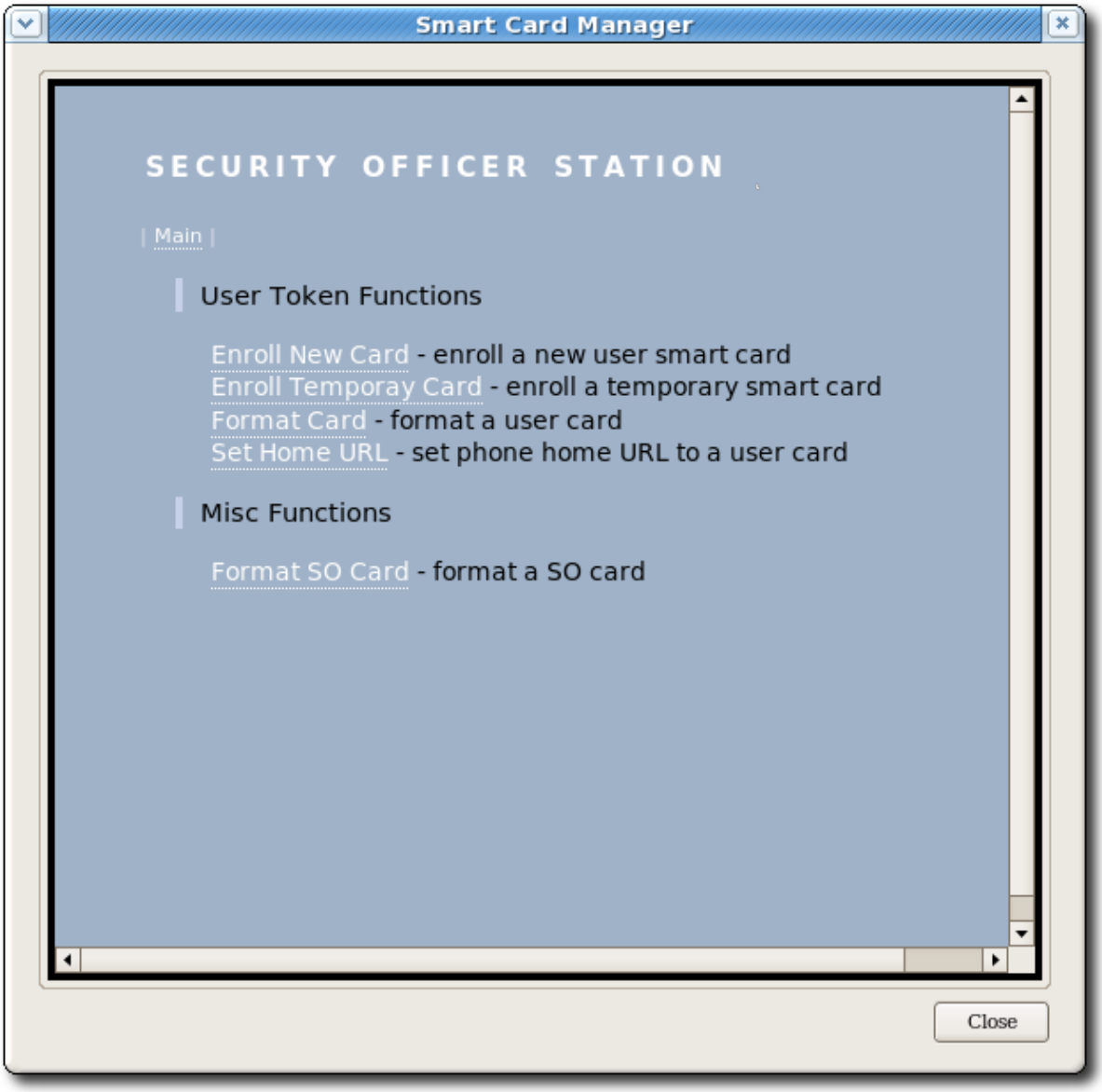
注記

コンピューターに接続したセキュリティー担当者カードが有効で登録されていることを確認します。以下のページにアクセスするには、セキュリティー担当者の認証情報が必要です。



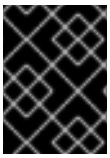
3. **Continue** をクリックすると、セキュリティー担当者の Station ページが表示されます。要求されたら、セキュリティー担当者のカードのパスワードを入力します。これは、SSL クライアント認証に必要です。

4. メニューから操作を選択します (一時トークンの登録、カードのフォーマット、または Phone Home URL の設定)。



5. *4章Enterprise security Client の設定* の説明に従って、操作を継続します。

4.5.3.3. 既存のセキュリティー担当者のスマートカードのフォーマット



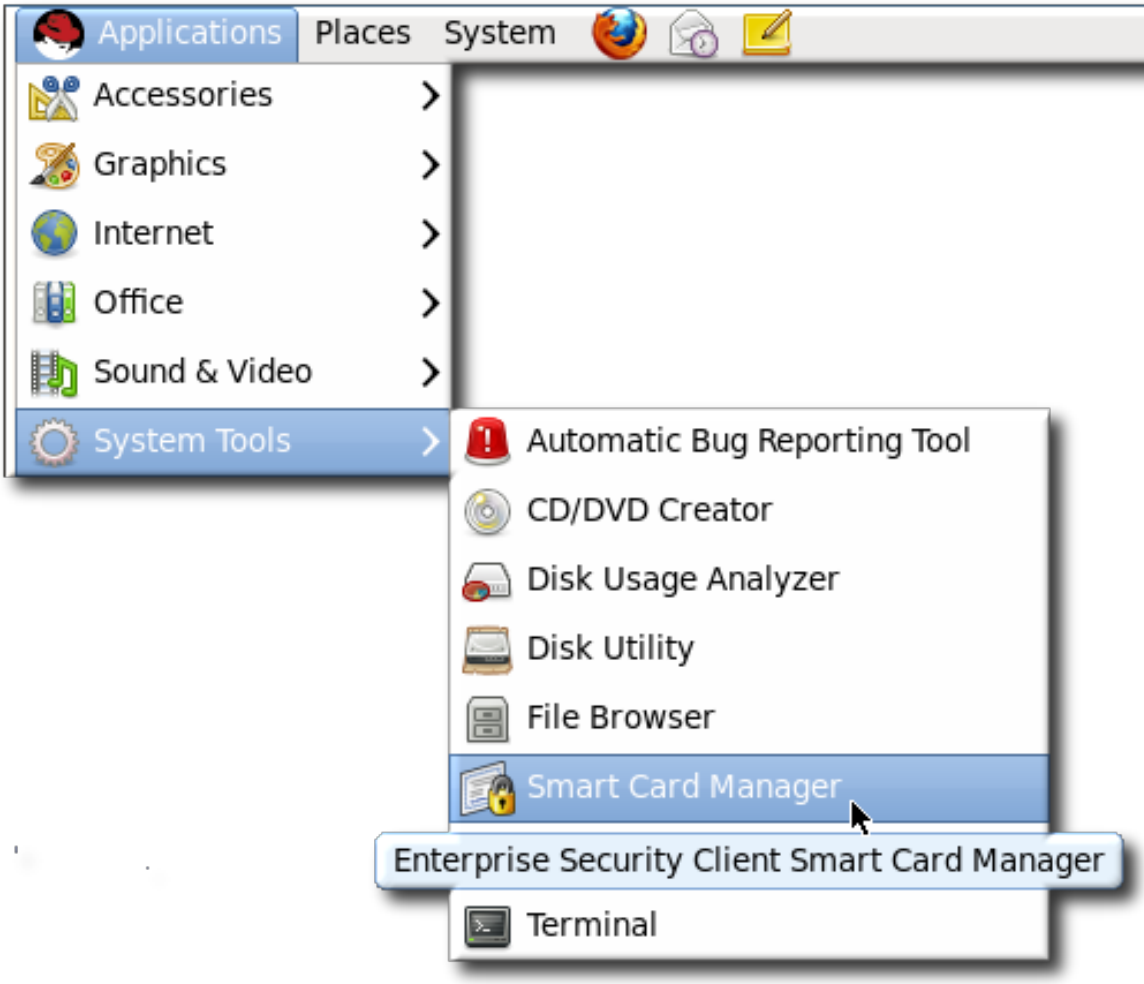
重要

トークンの再フォーマットは、セキュリティー担当者のトークンへの破壊的な操作であり、絶対に必要な場合のみ実行する必要があります。

1. 「セキュリティー担当者モードの有効化」 の説明に従って、セキュリティー担当者モードが有効になっていることを確認します。
2. Smart Card Manager UI を開きます。

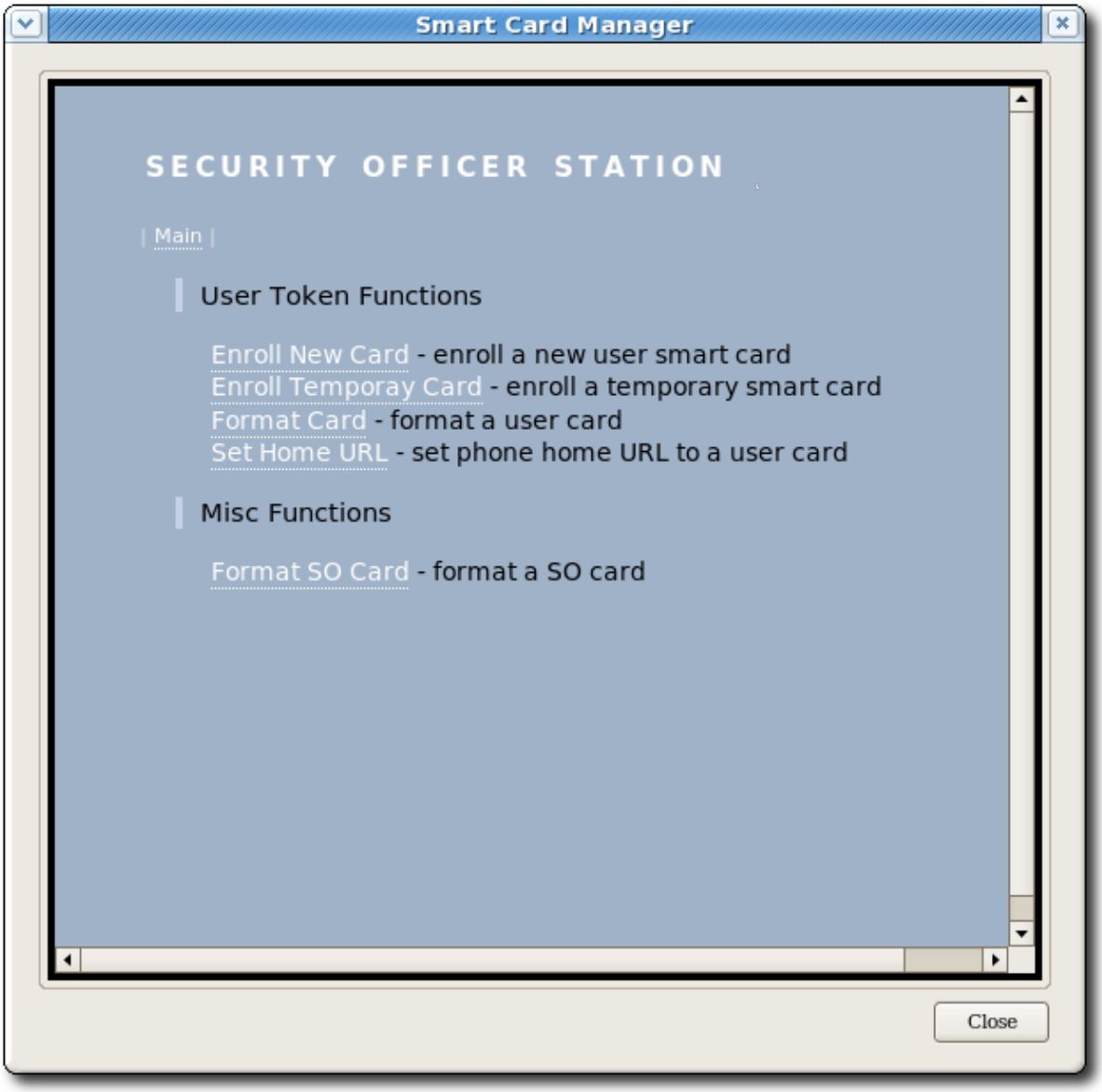
注記

コンピューターに接続したセキュリティー担当者カードが有効で登録されていることを確認します。以下のページにアクセスするには、セキュリティー担当者の認証情報が必要です。



3. **Continue** をクリックすると、セキュリティー担当者の Station ページが表示されます。要求されたら、セキュリティー担当者のカードのパスワードを入力します。これは、SSL クライアント認証に必要です。

4. メニューから操作を選択します (一時トークンの登録、カードのフォーマット、または Phone Home URL の設定)。



5. **Format SO Card** をクリックします。セキュリティー担当者のカードが既に挿入されているため、以下の画面が表示されます。

6. **Format** をクリックして操作を開始します。

カードが正常にフォーマットされると、セキュリティー担当者のカードの値がリセットされます。セキュリティー担当者のカードは、セキュリティー担当者モードに切り替え、さらなる操作を行うために使用する必要があります。

4.6. TPS を使用した SSL 接続の設定

デフォルトでは、TPS は標準の HTTP で Enterprise Security Client と通信します。多くの状況で好ましいことですが、HTTP over SSL (HTTPS) を使用して TPS クライアント通信をセキュアにすることも可能です。

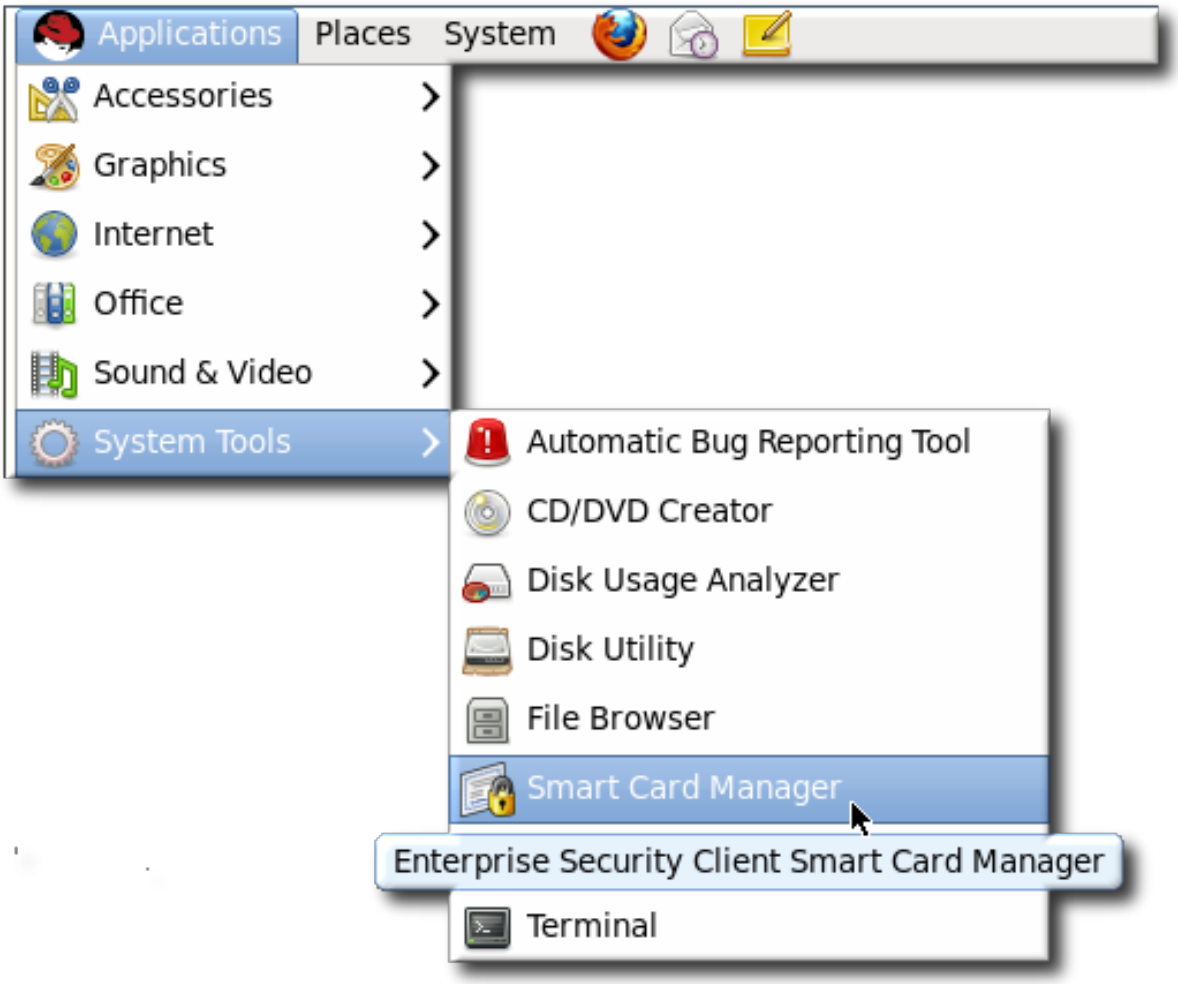
Enterprise Security Client には、TPS 接続を信頼するために TPS の証明書を発行した CA の CA 証明書が必要です。そこから、Enterprise Security Client を設定して TPS の SSL 証明書に接続できます。

1. TPS が使用する CA 証明書をダウンロードします。
    1. Web ブラウザーで CA のエンドユーザーページを開きます。

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

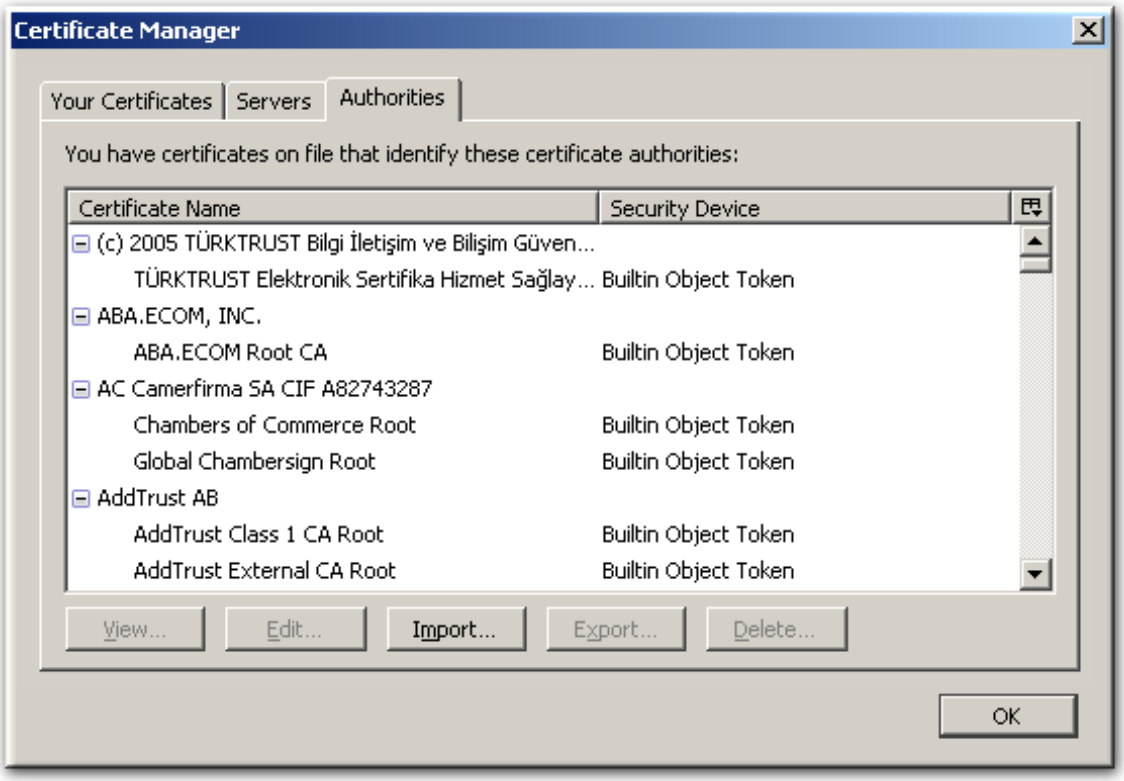
2. 上部の **Retrieval** タブをクリックします。
3. 左側のメニューで **Import CA Certificate Chain** リンクをクリックします。
4. チェーンをファイルとしてダウンロードするにはラジオボタンを選択し、ダウンロードしたファイルの場所と名前を書き留めます。

2. Enterprise Security Client を開きます。



3. CA 証明書をインポートします。
    1. **証明書の表示** ボタンをクリックします。

2. **認証** タブをクリックします。
3. **インポート** をクリックします。



4. CA 証明書チェーンファイルを参照し、これを選択します。
5. プロンプトが表示されたら、CA を信頼することを確認します。

4. Enterprise Security Client は、SSL 経由で TPS と通信するように設定する必要があります。これは、Enterprise Security Client が TPS の接続に使用するデフォルトである **Phone Home URL** を設定して行います。

5. 新しい空のトークンをマシンに挿入します。

   空のトークンはフォーマットされていないため、既存の Phone Home URL がないため、URL は手動で設定する必要があります。フォーマットされたトークン (トークンは製造元または IT 部門によってフォーマット可能) にはすでに URL が設定されているため、Phone Home URL の設定を求めるプロンプトは表示されません。

6. 新しい TPS URL に SSL ポート情報を入力します。以下に例を示します。

   ```
   https://server.example.com:7890/cgi-bin/home/index.cgi
   ```

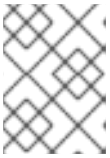
7. **テスト** ボタンをクリックして、TPS にメッセージを送信します。

   リクエストが正常に行われると、クライアントは Phone Home URL が正常に取得されたことを示すダイアログボックスを開きます。

4.7. スマートカード登録ユーザーインターフェースのカスタマイズ

TPS サブシステムは、一般的な形式のスマートカード登録画面を表示します。これは、初期化されていないスマートカードを挿入する際に自動的に開きます。これは、クライアントが稼働しているモードに応じて、実際には 3 つのページで構成されています。

- 通常の登録の **/var/lib/pki-tps/cgi-bin/home/Enroll.html**
- セキュリティー担当者の登録の **/var/lib/pki-tps/cgi-bin/so/Enroll.html**
- セキュリティー担当者ワークステーション登録 (セキュリティー担当者 UI から登録されたユーザー) の **/var/lib/pki-tps/cgi-bin/sow/Enroll.html**



注記

セキュリティー担当者ワークステーションディレクトリーには、形式や PIN リセットなどのその他のトークン操作用の HTML ファイルが含まれます。

カスタムユーザープロファイルがある場合は、さらに多くの登録ページがある可能性があります。

これらの登録ページは基本的な HTML および JavaScript で、外観と機能を簡単にカスタマイズできます。登録ファイルによって参照されるイメージや JavaScript ファイルなどのリソースは、**/var/lib/pki-tps/cgi-bin/sow** のセキュリティー担当者の登録ファイルの **/var/lib/pki-tps/docroot/esc/sow** など、対応する **docroot/** ディレクトリーにあります。

スマートカードの登録ページをカスタマイズする方法は複数あります。最初の、および最も簡単なものは、ページのテキストを変更します。ページタイトル、セクション見出し、フィールド名、および説明はすべて、例4.3「ページテキストの変更」 の抽出に示したように HTML ファイルを編集して変更できます。

例4.3 ページテキストの変更

```
<!-- Change the title if desired -->
<title>Enrollment</title>
```

```
...
<p class="headerText">Smartcard Enrollment</p>
...
<!-- Insert customized descriptive text here. -->
<p class="bodyText">You have plugged in your smart card!
        After answering a few easy questions, you will be able to use your smart card.
</p>
<p class="bodyText">
       Now we would like you to identify yourself.
</p>
...
<table>
    <tr>
        <td><p >LDAP User ID: </p></td>
        <td> </td>
        <td><input type="text" id="snametf" value=""></td>
     </tr>
</table>
```

ページのスタイルは、**style.css** CSS スタイルシートとロゴイメージ **logo.png** の 2 つのファイルで変更できます。

例4.4 ページスタイルの変更

```
<link rel=stylesheet href="/esc/home/style.css" type="text/css">
...

<table width="100%" class="logobar">
  <tr>
    <td>
<img alt="" src="/home/logo.jpg">
    </td>
    <td>
      <p class="headerText">Smartcard Enrollment</p>
    </td>
  </tr>
</table>
```

この **style.css** ファイルは、標準の CSS ファイルであるため、タグとクラスはすべて以下のように定義できます。

```
body {
background-color: grey;
       font-family: arial;
       font-size: 7p
}
```

CSS の詳細は、http://www.w3.org/Style/CSS/learning で参照できます.。

Enroll.html ファイルをカスタマイズする最後の手段では、ページ機能を設定する JavaScript ファイルを使用します。このファイルは、進捗メーターなどの機能や、ユーザーディレクトリーに対するユーザーの認証に使用される入力の処理を制御します。

例4.5 ページスクリプトの変更

```
<progressmeter id="progress-id" hidden="true" align = "center"/>
...
<table>
    <tr>
        <td><p >LDAP User ID: </p></td>
        <td> </td>
        <td><input type="text" id="snametf" value=""></td>
     </tr>
</table>
```

警告

util.js ファイルの変更は十分に注意してください。このファイルを誤って編集した場合は、Enterprise Security Client UI が破損し、トークンが登録できなくなります。

完全な **/var/lib/pki-tps/cgi-bin/home/Enroll.html** ファイルは例4.6「完全な登録 .html ファイル」 にあります。

例4.6 完全な登録 .html ファイル

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel=stylesheet href="/esc/home/style.css" type="text/css">

<title>Enrollment</title>
</head>
<script type="text/JavaScript" src="/esc/home/util.js">
</script>
<body onload="InitializeBindingTable();" onunload=cleanup()>

<progressmeter id="progress-id" hidden="true" align = "center"/>
<table width="100%" class="logobar">
  <tr>
    <td>
<img alt="" src="/home/logo.jpg">
    </td>
    <td>
      <p class="headerText">Smartcard Enrollment</p>p
    </td>
  </tr>
</table>
  <table id="BindingTable" width="200px"align="center">
    <tr id="HeaderRow">
    </tr>
```

```
  </table>
  <p class="bodyText">You have plugged in your smart card! After answering a few easy
questions, you will be able to use your smart card.
  </p>p
  <p class="bodyText">
      Now we would like you to identify yourself.
   </p>p
    <table>
      <tr>
        <td><p >LDAP User ID: </p>p</td>
        <td> </td>
        <td><input type="text" id="snametf"  value=""></td>
         <td> </td>
        <td><p>LDAP Password: </p>p</td>
         <td> </td>
        <td><input type="password" id="snamepwd" value=""></td>
      </tr>

      </table>

      <p class="bodyText"> Before you can use your smart card, you will need a password to
protect it.</p>p
      <table>
      <tr>
        <td><p >Password:</p>p</td>
        <td><input type="password" id="pintf" name="pintf" value=""></td>

        <td><p >Re-Enter Password:</p>p</td>
        <td><input type="password" id="reenterpintf" name="reenterpintf" value=""></td>
    </table>
    <br>
    <table width="100%">
      <tr>
        <td align="right">
          <input type="button" id="enrollbtn" name="enrollbtn" value="Enroll My Smartcard"
onClick="DoEnrollCOOLKey();">
        </td>
      </tr>
    </table>
</body></html>
```

4.8. トークン操作の LDAP 認証の無効化

デフォルトでは、トークン操作を要求する各ユーザーは LDAP ディレクトリーに対して認証されます。ユーザーにエントリーがある場合、操作が許可されます。ユーザーにエントリーがない場合、操作は拒否されます。

テストまたは特定タイプのユーザーの場合、LDAP 認証を無効にする方が単純で、より簡単で適切な方法になります。これは、Enterprise Security Client 設定で設定されませんが、トークン処理システム設定で設定されるため、TPS 管理者によって実行する必要があります。

1. TPS サブシステムを停止します。

```
service pki-tps stop
```

2. TPS 設定ファイルを開きます。

```
vim /var/lib/pki-tps/conf/CS.cfg
```

3. 認証パラメーターを **false** に設定します。

```
op.operation_type.token_type.loginRequest.enable=false
op.operation_type.token_type.auth.enable=false
```

operation_type は、**enroll**、**format**、または **pinreset** など、LDAP 認証が無効化されているトークン操作です。ある操作タイプで認証を無効にしても、他の操作タイプでは無効になりません。

token_type は、トークンプロファイルです。通常のユーザー、セキュリティー担当者、およびセキュリティー担当者が登録したユーザーには、デフォルトのプロファイルがあります。他の種類のユーザーや証明書のカスタムトークンタイプもあります。

以下に例を示します。

```
op.enroll.userKey.loginRequest.enable=false
op.enroll.userKey.pinReset.enable=false
```

4. TPS サブシステムを再起動します。

```
service pki-tps start
```

TPS 設定の編集については、『Certificate System Administrator's Guide』 で説明されています。

第5章 ENTERPRISE SECURITY CLIENT でのスマートカードの使用

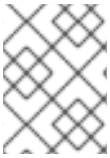
スマートカードが**登録**されると、ユーザー固有の鍵と証明書が生成され、カードに格納されます。Red Hat Enterprise Linux では、ユーザーと、証明書を発行するシステムの間で機能するインターフェースは、**Enterprise Security Client** です。Enterprise Security Client は、スマートカードが挿入された (または削除される) 時に認識され、Red Hat Certificate System の適切なサブシステムに通知します。その後、そのサブシステムが証明書資料を生成し、それらを Enterprise Security Client に送信し、トークンに書き込みます。登録プロセスです。

以下のセクションでは、Enterprise Security Client を使用したトークンの登録、フォーマット、およびパスワードのリセット操作に関する基本的な手順を説明します。

5.1. 対応するスマートカード

Enterprise Security Client は、JavaCard 2.1 以降および Global Platform 2.01 準拠のスマートカードをサポートし、以下のカードを使用してテストされました。

- SafeNet 330J Java スマートカード
- Gemalto 64K V2 トークン (スマートカードおよび GemPCKey USB フォームファクターキーの両方)
- Gemalto GCx4 72K および TOPDLGX4 144K 共通アクセスカード (CAC)
- Ovyhur ID 1 V5.2 Common Access Card (CAC)
- FIPS 201 に準拠したPIV (Personal Identity Verification) カード



注記

Enterprise Security Client は PIV カードまたは CAC カードをプロビジョニングしませんが、PIV カードを読み込んで情報を表示します。

スマートカードテストでは、2 枚のカードリーダーを使用していました。

- SCM SCR331 CCID
- OMNIKEY 3121

Enterprise Security Client でサポートされている唯一のカードマネージャーアプレットは CoolKey アプレットです。

5.2. 登録するユーザーの設定

Token Processing System がインストールされると、設定の 1 つに、トークンの登録が許可されるユーザーが含まれる LDAP ディレクトリーがあります。この認証ディレクトリーに保存されているユーザーのみが、登録、フォーマット、またはトークンを持つことができます。トークンまたはスマートカードの登録を試みる前に、操作を要求するすべてのユーザーの LDAP ディレクトリーにエントリーがあることを確認してください。

TPS は、LDAP ディレクトリーの特定のベース DN を確認するように設定されています。これは TPS の **CS.cfg** で設定されます。

```
auth.instance.0.baseDN=dc=example,dc=com
auth.instance.0.hostport=server.example.com:389
```

ユーザーによるトークンの登録を許可するには、ユーザーはベース DN の背後にある必要があります。

ユーザーにエントリーがない場合は、ユーザーにトークンを登録する前に、指定したベース DN の指定された LDAP ディレクトリーに管理者がユーザーを追加する必要があります。

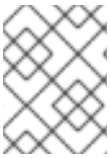
```
/usr/bin/ldapmodify -a -D "cn=Directory Manager" -w secret -p 389 -h server.example.com

dn: uid=jsmith,ou=People, dc=example,dc=com
objectclass: person
objectclass: inetorgperson
objectclass: top
uid: jsmith
cn: John Smith
email: jsmith@example.com
userPassword: secret
```

5.3. スマートカードの自動登録

Enterprse Security Client は Phone Home 機能を使用して設定されるため、スマートカードの登録は非常に簡単です。バックエンド TPS サーバーへの接続に必要な情報は各スマートカードで提供されます。そのため、ユーザーは手順を迅速かつ簡単に実施することができます。

初期化されていないスマートカードを登録するには、以下を実行します。



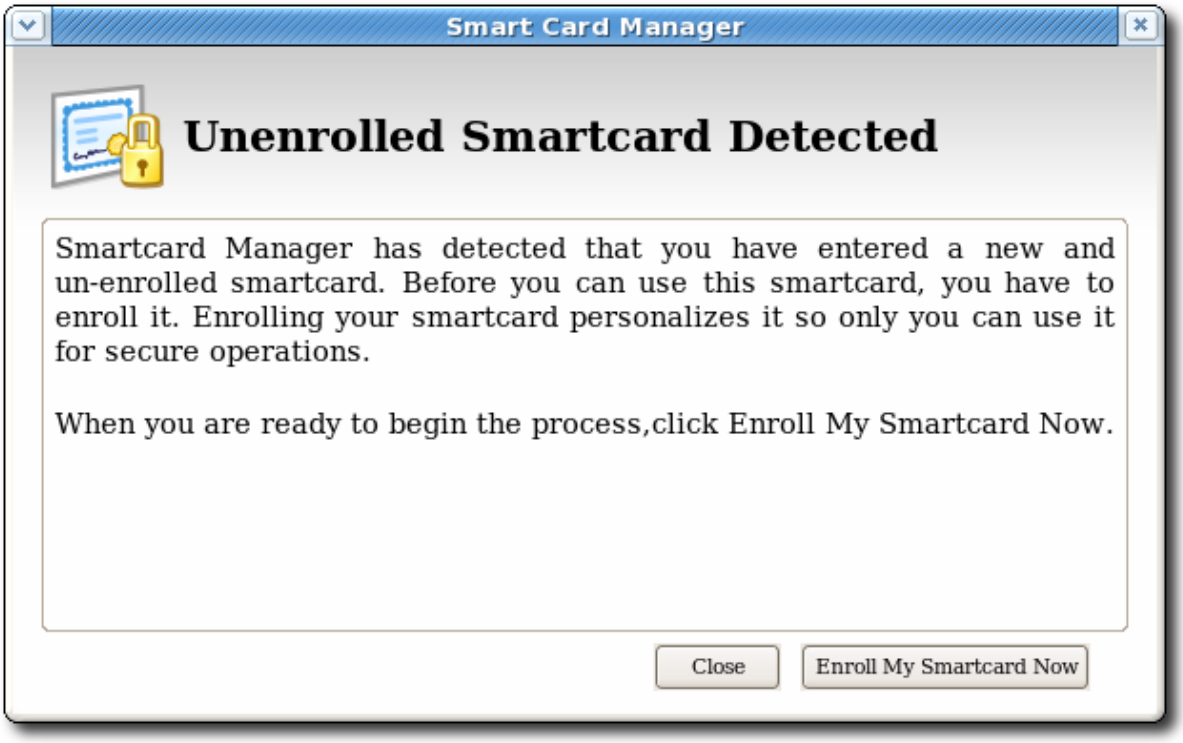
注記

この手順では、スマートカードが未初期化で、適切な Phone Home 情報が設定されていることを前提としています。

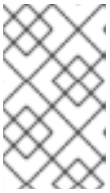
1. Enterprise Security Client が実行されていることを確認します。
2. 初期化されていないスマートカードを挿入します。このカードは、ユーザーの組織の TPS および登録インターフェース URL の Phone Home 情報で事前にフォーマットされているものとします。

   スマートカードは、USB フォームファクタースマートカードを空き USB スロットに配置するか、または標準のフルサイズのスマートカードをスマートカードリーダーに挿入して追加できます。

   システムがスマートカードを認識すると、初期化されていないスマートカードを検出したことを示すメッセージが表示されます。



3. **Enroll My Smart Card Now** をクリックして、スマートカードの登録フォームを表示します。

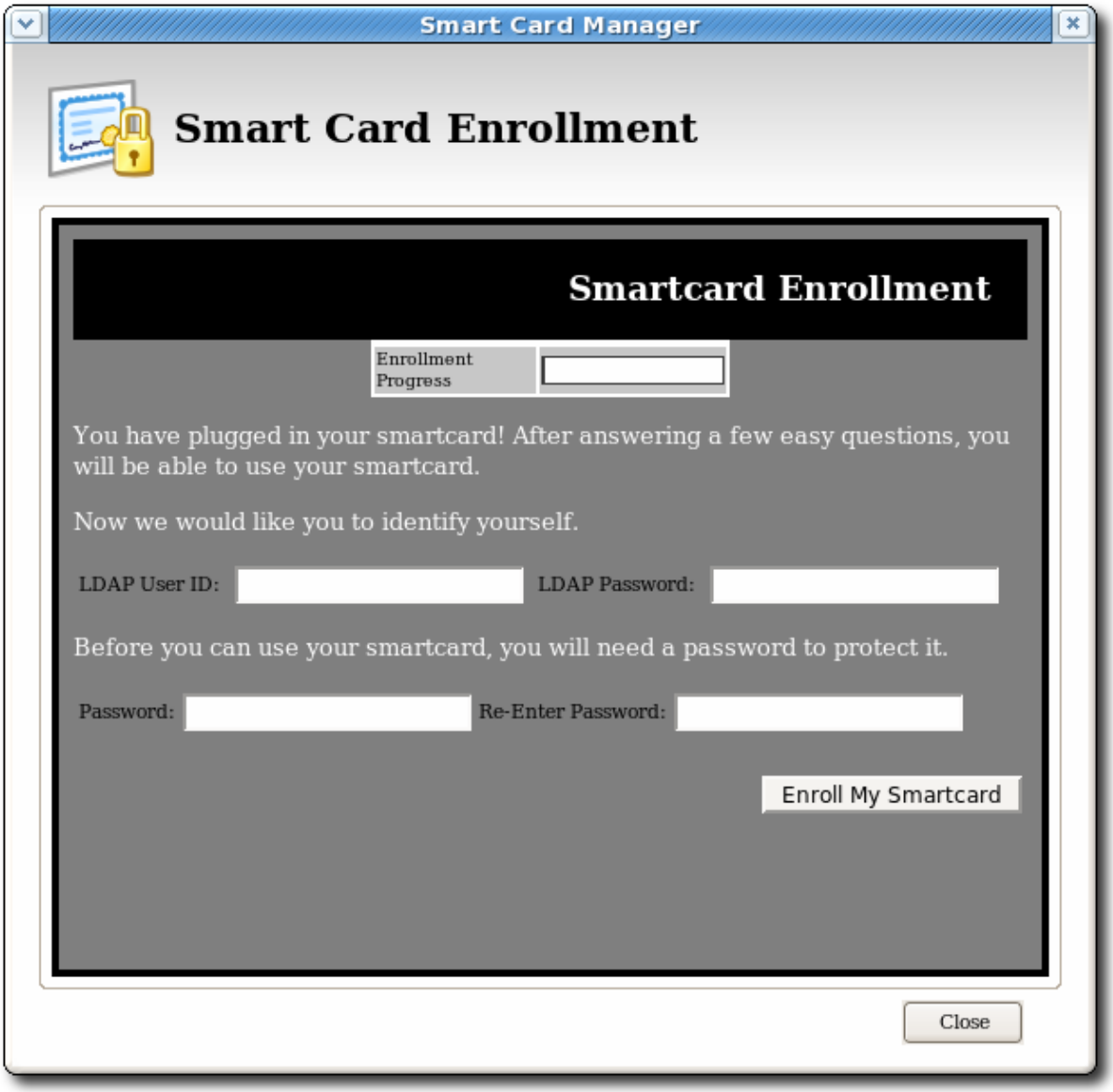


注記

この時点でカードを削除すると、スマートカードが検出されなくなったことを示すメッセージが表示されます。登録プロセスを続行するためにカードを再挿入します。

登録ファイルはリモートでアクセスされ、TPS インスタンスにあります。ネットワーク接続が不良または破損している場合は、**ネットワーク接続を確認して再試行**というメッセイジーが表示されることがあります。登録ウィンドウを開くこともできますが、登録プロセスは続行されません。登録ページは、Enterprise Security Client が正常に接続されている場合はキャッシュできるため、ネットワークがオフラインであっても登録 UI が開きます。Enterprise Security Client を再起動して、ネットワーク接続を確認します。

4. Smart Card Manager は、登録 UI がどこにあるかを認識するようになりました (これは、Phone Home 情報に含まれています)。ユーザーが必要な情報を入力するための登録フォームが表示されます。



この図では、TPS サーバーに含まれるデフォルトの登録 UI を示しています。この UI は標準の HTML 形式で、独自のデプロイメント要件に合わせてカスタマイズできます。これには、企業のロゴの追加や、フィールドテキストの追加および変更が含まれます。

UI のカスタマイズに関する情報は、「スマートカード登録ユーザーインターフェースのカスタマイズ」 を参照してください。

5. TPS サーバーがスマートカードの登録操作を処理するには、サンプル登録 UI に以下の情報が必要です。

    - **LDAP ユーザー ID**これは、スマートカードを登録するユーザーの LDAP ユーザー ID です。画面名または従業員またはカスタマー ID 番号を指定することもできます。

    - **LDAP パスワード**これは、入力したユーザー ID に対応するパスワードです。簡単なパスワードまたは顧客番号を指定できます。

注記

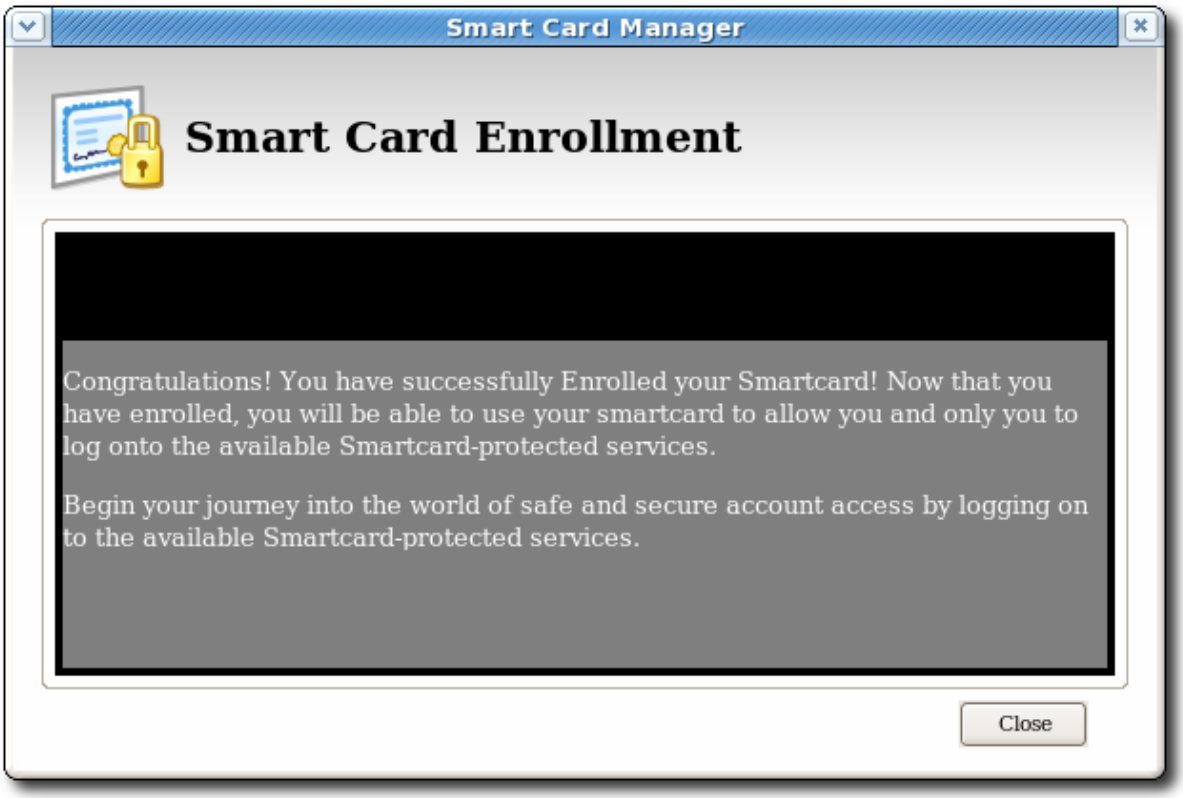
LDAP ユーザー ID およびパスワードは Directory Server ユーザーに関連するものです。TPS サーバーは、通常、ユーザー情報と TPS がユーザーを認証する Directory Server に関連付けられます。

パスワードは、Directory Server で設定したパスワードポリシーに準拠する必要があります。

- **パスワードとパスワードの再入力**これらのフィールドは、カード情報を保護するために使用されるスマートカードのパスワードを設定し、確認します。

6. 必要な情報をすべて入力したら、**Enroll My Smart Card** をクリックして情報を送信し、カードを登録します。

7. 登録プロセスが完了すると、カードが正常に登録されたことを示すメッセージページが開き、新たに登録したスマートカードを使用するカスタム命令が提供されます。



5.4. スマートカードの管理

Manage Smart Cards ページを使用すると、トークンに保存されている暗号鍵のいずれかに適用できる多くの操作を実行できます。

このページを使用してトークンのフォーマットし、カードのパスワードの設定およびリセットして、カード情報の表示を行うことができます。その他 2 つの操作 (トークンの登録および診断ログの表示) は、**Manage Smart Cards** ページからもアクセスできます。これらの操作は他のセクションで扱われます。

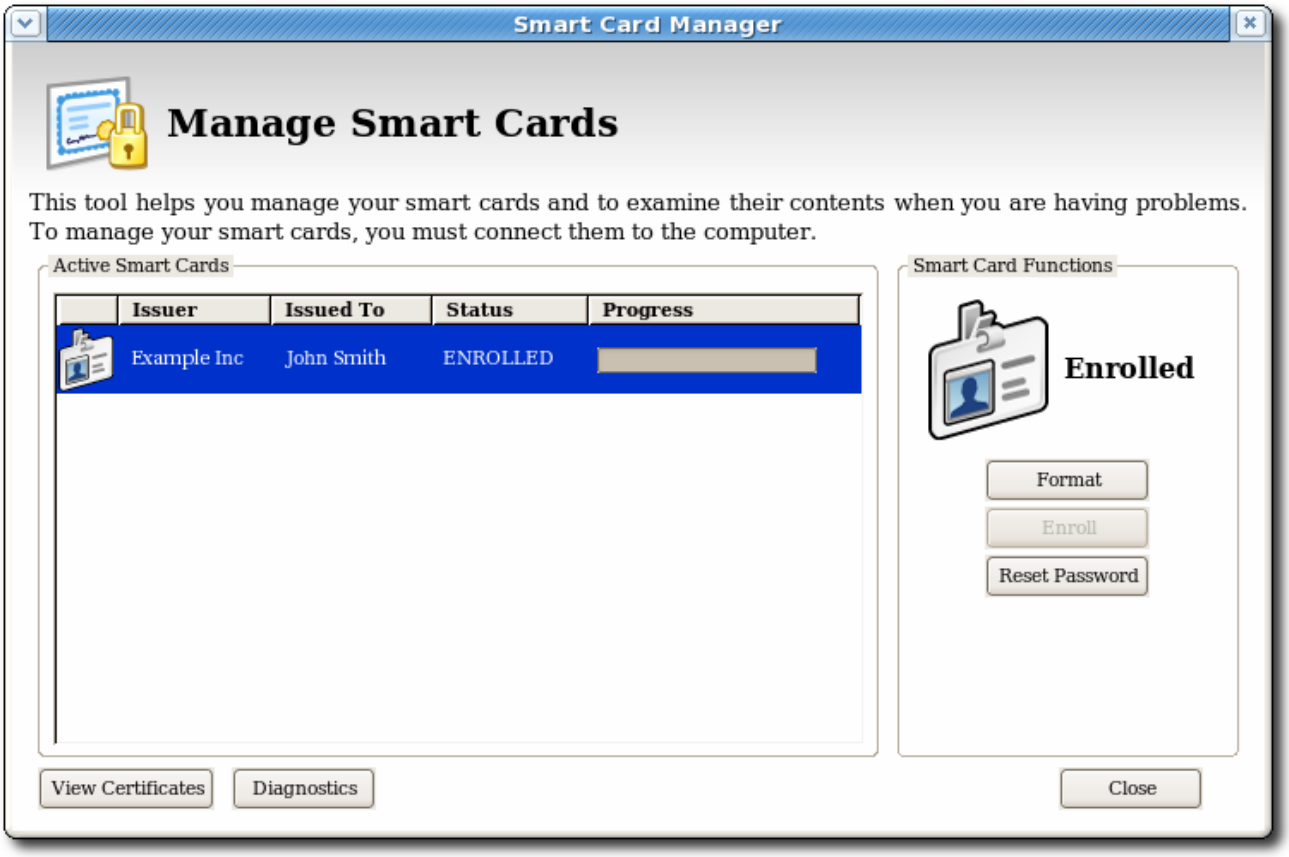


図5.1 Manage Smart Cards ページ

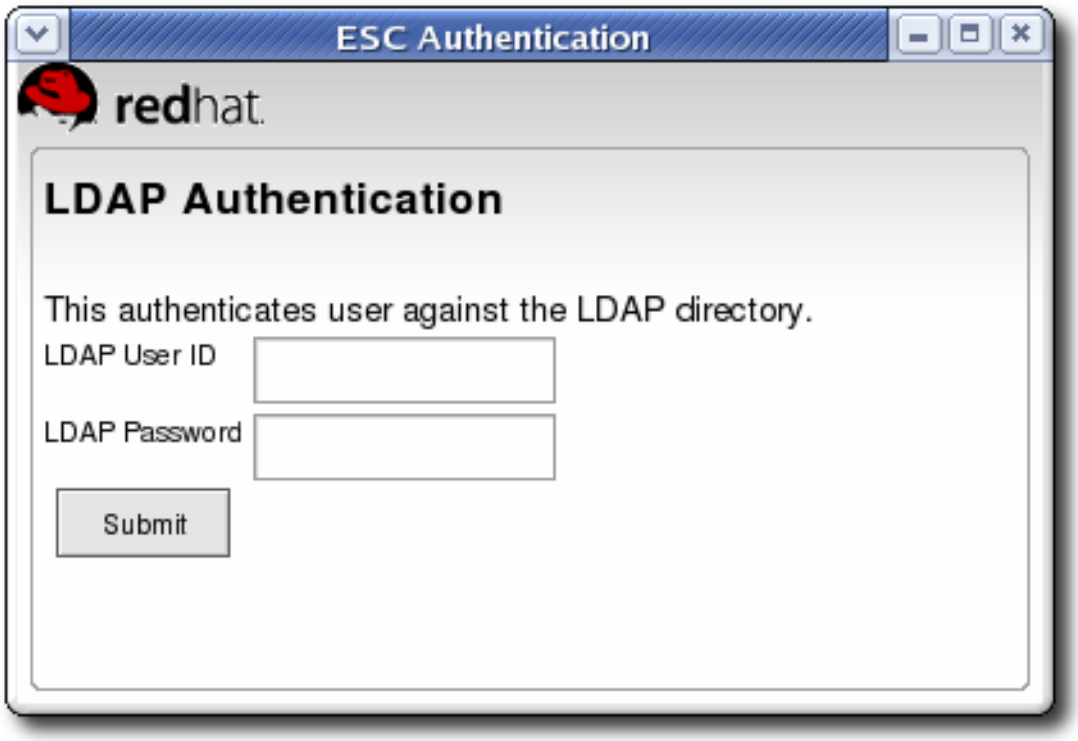
5.4.1. スマートカードのフォーマット

スマートカードをフォーマットすると、初期化されていない状態にリセットされます。これにより、以前に生成されたユーザー鍵のペアがすべて削除され、登録中にスマートカードに設定されたパスワードが消去されます。

TPS サーバーは、新しいバージョンのアプレット鍵と対称キーをカードにロードするように設定できます。TPS は、Red Hat Enterprise Linux 6 に同梱される CoolKey アプレットをサポートします。

スマートカードのフォーマット:

1. 対応しているスマートカードをコンピューターに挿入します。カードが **Active Smart Cards** テーブルに表示されることを確認します。
2. **Manage Smart Cards** 画面の **Smart Card Functions** セクションで、**Format** をクリックします。
3. TPS がユーザー認証用に設定されている場合は、認証ダイアログにユーザー認証情報を入力して、**Submit** をクリックします。



4. フォーマット処理中に、カードのステータスが BUSY に変更され、進捗バーが表示されます。フォーマットプロセスが完了すると、成功メッセージが表示されます。**OK** をクリックしてメッセージボックスを閉じます。

5. フォーマットプロセスが完了すると、**Active Smart Cards** の表に、UNINITIALIZED というカードステータスが表示されます。

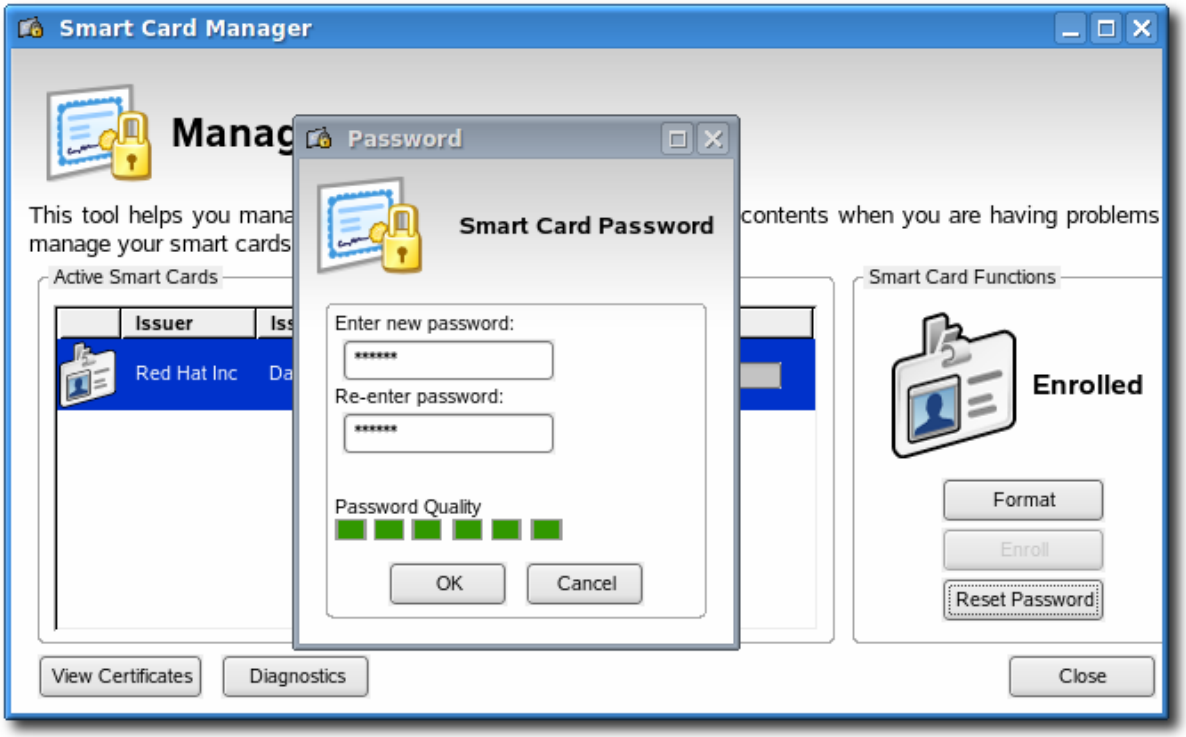
5.4.2. スマートカードパスワードのリセット

1. 対応しているスマートカードをコンピューターに挿入します。カードが **Active Smart Cards** テーブルに表示されることを確認します。

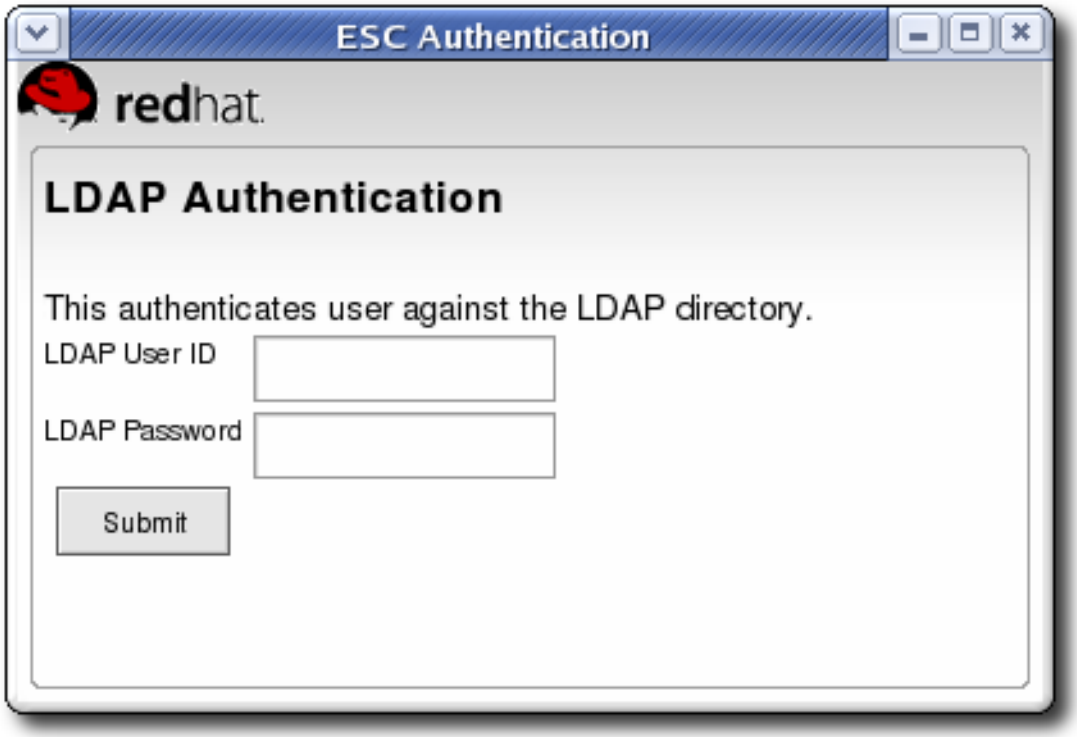
2. Manage Smart Cards 画面の **Smart Card Functions** セクションで、**Reset Password** をクリックして、**Password** ダイアログを表示します。

3. **Enter new password** フィールドに新しいスマートカードパスワードを入力します。

4. **Re-Enter password** フィールドで新しいスマートカードパスワードを確認して、**OK** をクリックします。



5. TPS がユーザー認証用に設定されている場合は、認証ダイアログにユーザー認証情報を入力して、**Submit** をクリックします。



6. パスワードのリセットが終了するのを待ちます。

5.4.3. 証明書の更新

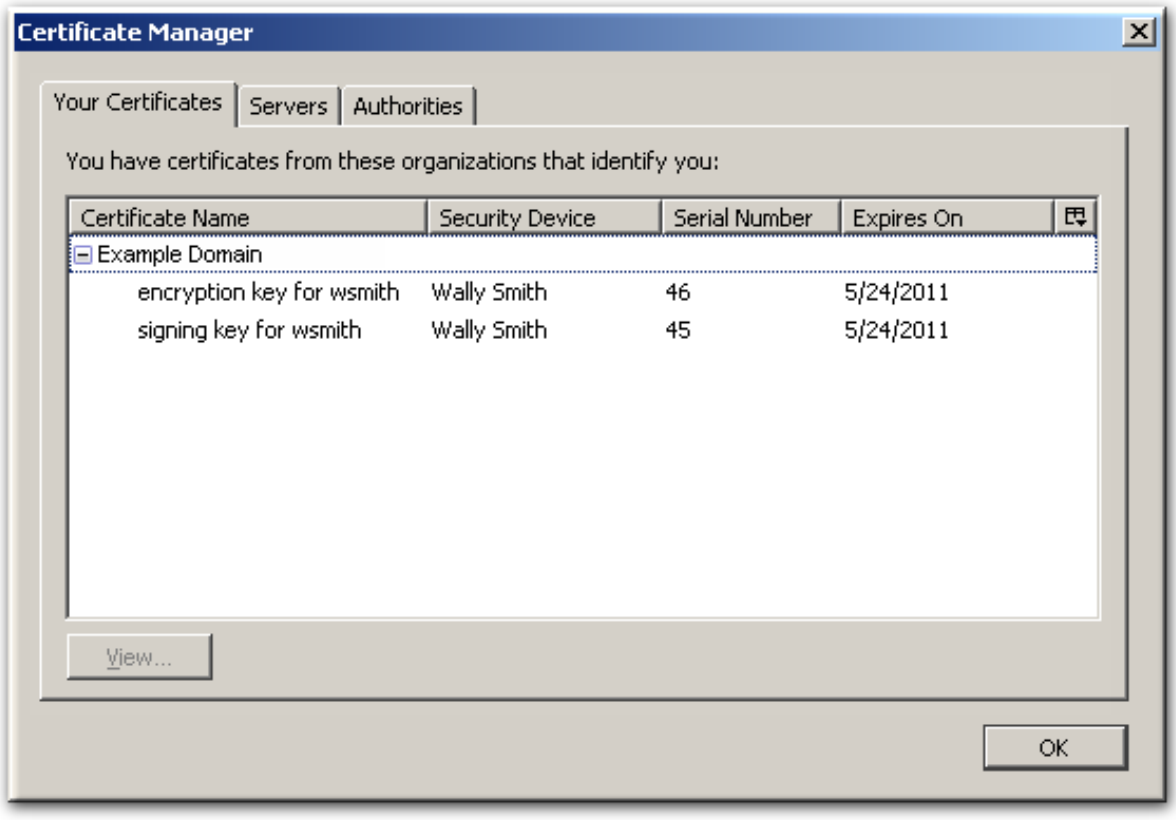
Smart Card Manager は、保存した鍵や証明書など、選択したスマートカードの基本情報を表示できます。証明書情報を表示するには、以下を実行します。

1. 対応しているスマートカードをコンピューターに挿入します。カードが **Active Smart Cards** テーブルに表示されることを確認します。

2. 一覧からカードを選択し、**View Certificates** をクリックします。

これにより、シリアル番号、証明書のニックネーム、および有効日など、カードに保存されている証明書の基本情報を表示します。

3. 証明書に関する詳細情報を表示するには、一覧から証明書を選択して **表示** をクリックします。



5.4.4. CA 証明書のインポート

XULRunner Gecko エンジンは、ブラウザーまたは Enterprise Security Client のようにクライアントが表示する SSL ベースの URL に対する文字列制御を実装します。(XULRunner フレームワーク経由の Enterprise Security Client が URL を信頼しない場合、その URL はアクセスできません。

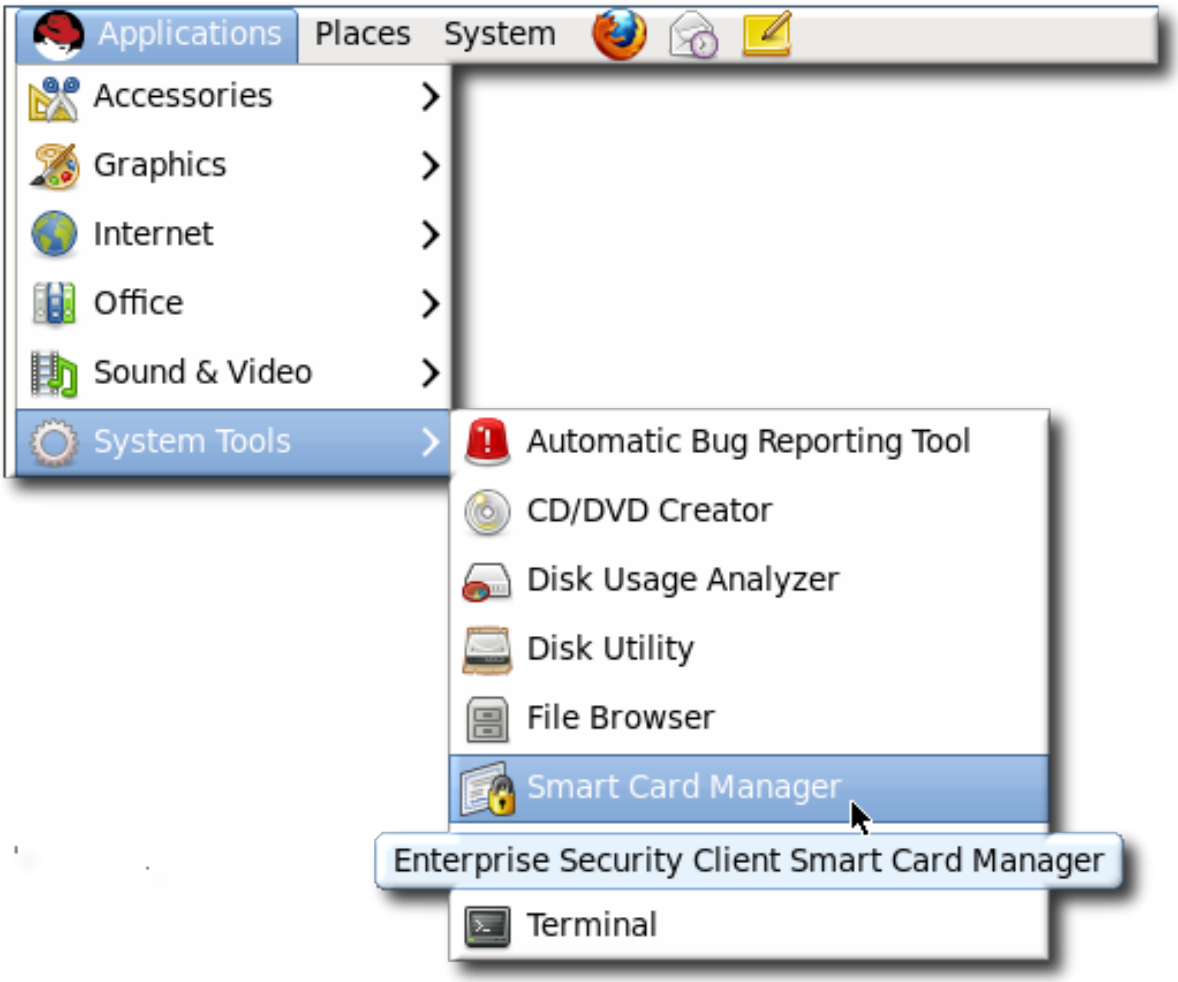
SSL ベースの URL を信頼する 1 つの方法として、サイトの証明書を発行した CA の CA 証明書チェーンをインポートおよび信頼する方法があります。(もう 1 つは、「サーバーの例外の追加」 にあるように、そのサイトに信頼**セキュリティー例外**を作成することです。)

スマートカードの証明書を発行する CA は、Enterprise Security Client アプリケーションによって信頼される必要があります。つまり、その CA 証明書を Enterprise Security Client にインポートする必要があります。

1. Web ブラウザーで CA のエンドユーザーページを開きます。

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

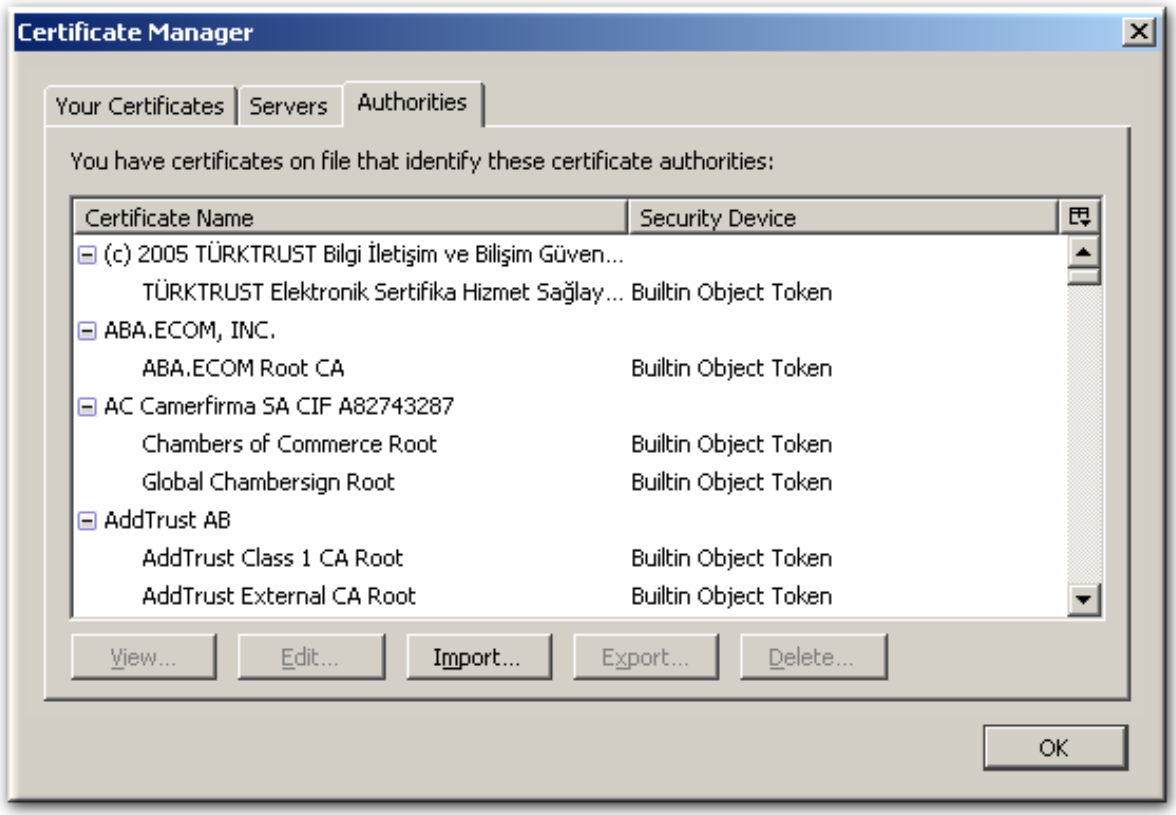
2. 上部の **Retrieval** タブをクリックします。
3. 左側のメニューで **Import CA Certificate Chain** リンクをクリックします。
4. チェーンをファイルとしてダウンロードするにはラジオボタンを選択し、ダウンロードしたファイルの場所と名前を書き留めます。
5. スマートカードマネージャー GUI を開きます。



6. **証明書の表示** ボタンをクリックします。

7. **認証** タブをクリックします。

8. **インポート** をクリックします。



9. CA 証明書チェーンファイルを参照し、これを選択します。

10. プロンプトが表示されたら、CA を信頼することを確認します。

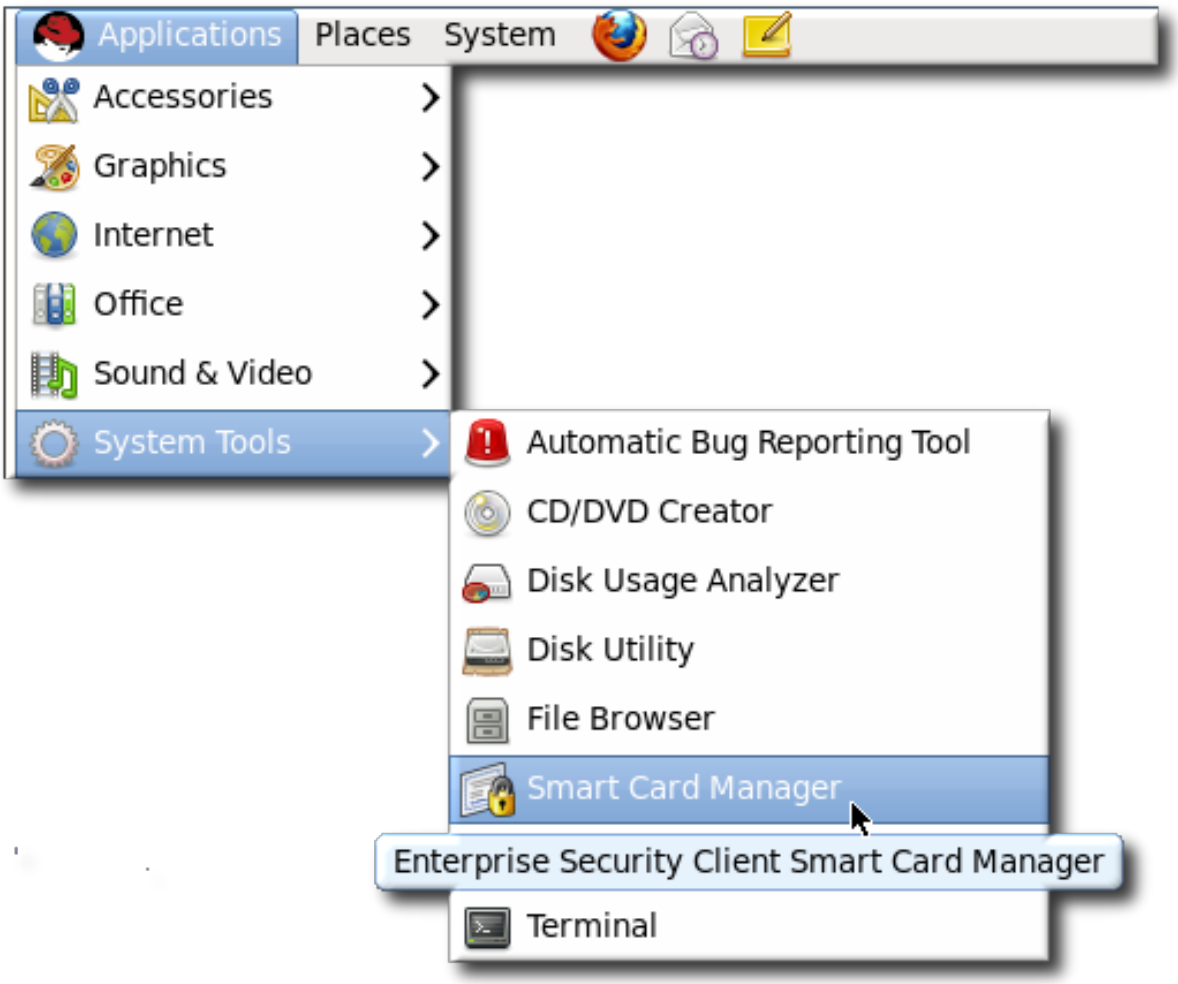
5.4.5. サーバーの例外の追加

XULRunner Gecko エンジンは、ブラウザーまたは Enterprise Security Client のようにクライアントが表示する SSL ベースの URL に対する文字列制御を実装します。(XULRunner フレームワーク経由の Enterprise Security Client が URL を信頼しない場合、その URL はアクセスできません。

SSL ベースの URL を信頼する 1 つの方法として、サイトの証明書をインポートして、Enterprise Security Client がそれを認識するよう強制するサイトに信頼 **セキュリティー例外** を作成するものがあります。(他に、「CA 証明書のインポート」 にあるように、サイトの CA 証明書チェーンをインポートし、自動的に信頼するオプションがあります。)

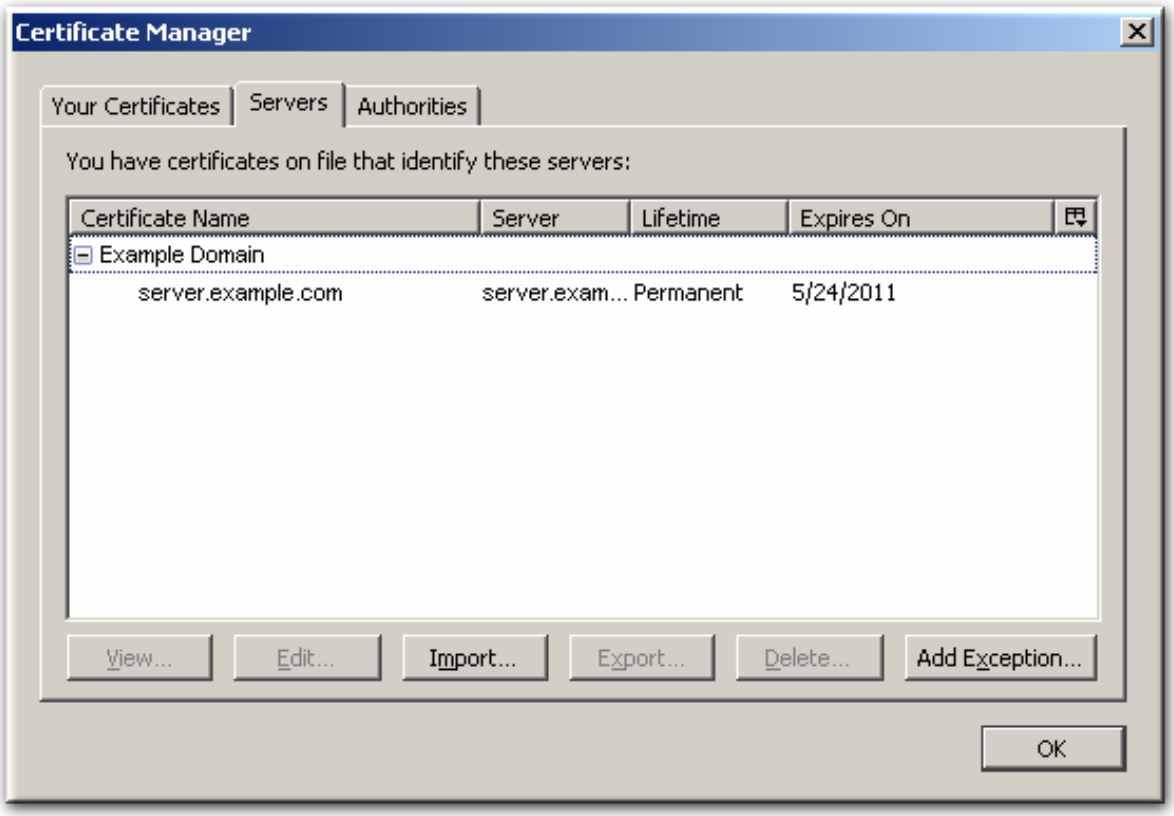
スマートカードは、特別なセキュリティー例外を必要とする SSL 経由でサービスまたは Web サイトにアクセスするために使用できます。このような例外は、Mozilla Firefox などのブラウザーで Web サイトに例外を設定するのと同様に、Enterprise Security Client で設定できます。

1. Smart Card Manager UI を開きます。

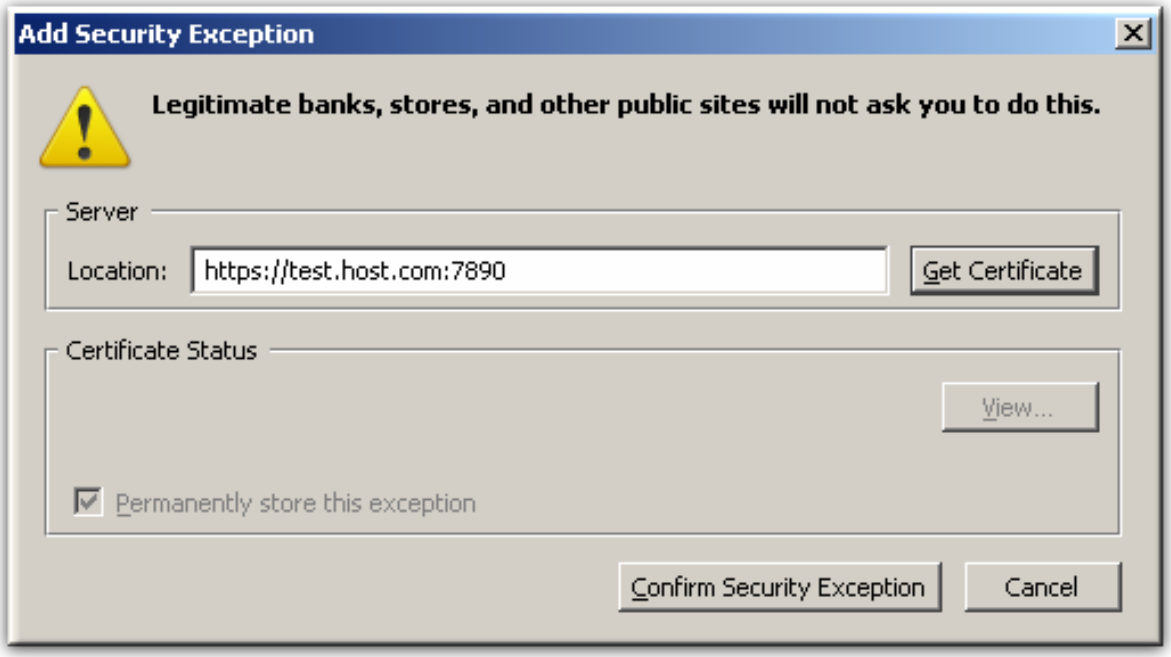


2. **証明書の表示** ボタンをクリックします。

3. **Servers** タブをクリックします。
4. **Add Exception** をクリックします。



5. スマートカードがアクセスに使用されるサイトまたはサービスの URL (ポート番号) を入力します。次に、**証明書の取得** ボタンをクリックして、サイトのサーバー証明書をダウンロードします。

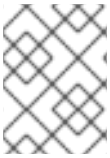


6. **セキュリティー例外の確認**をクリックして、許可されたサイトの一覧にサイトを追加します。

5.4.6. スマートカードの登録

ほとんどのスマートカードは、「スマートカードの自動登録」 に説明されている自動登録手順を使用して自動的に登録されます。**Manage Smart Cards** 機能を使用して、スマートカードを手動で登録することもできます。

ユーザーキーペアでトークンを登録する場合、トークンは SSL クライアント認証や S/MIME などの証明書ベースの操作に使用できます。



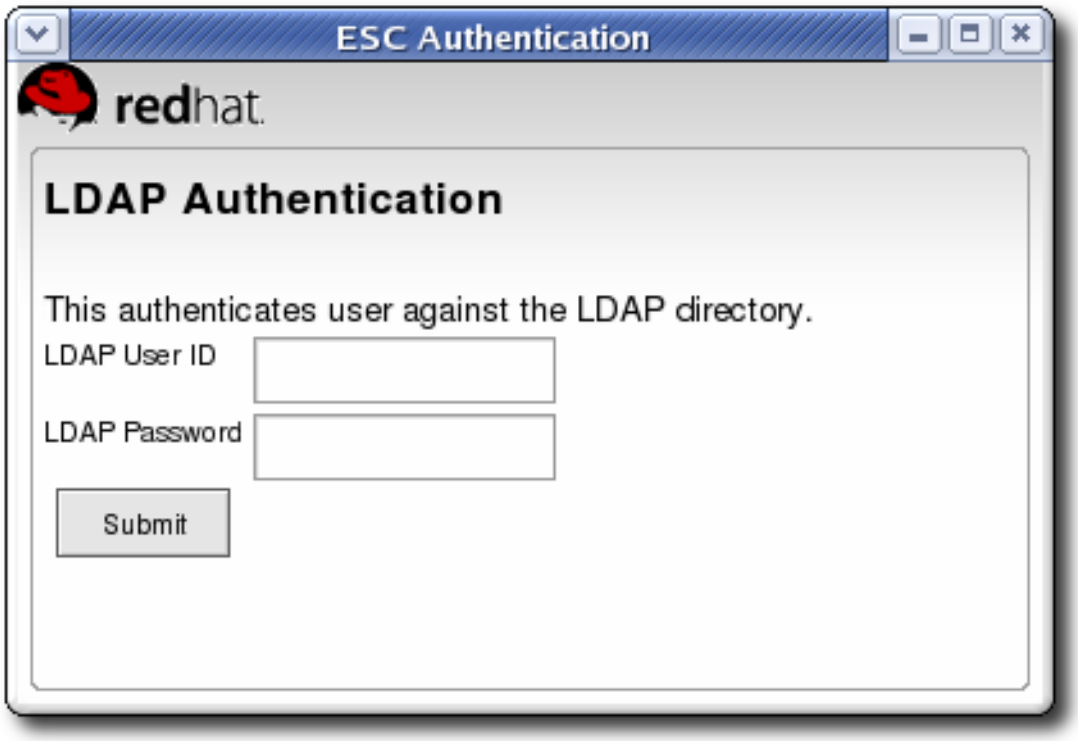
注記

TPS サーバーは、トークンが失われた場合のリカバリーのために、サーバー上でユーザーキーペアを生成し、DRM サブシステムでアーカイブするように設定できます。

スマートカードを手動で登録するには、以下を実行します。

1. 対応している未登録のスマートカードをコンピューターに挿入します。カードが **Active Smart Cards** テーブルに表示されることを確認します。
2. **Enroll** をクリックして、**Password** ダイアログを表示します。
3. **Enter a password** フィールドに新しいキーパスワードを入力します。

   Re-Enter a password フィールドで新規パスワードを確認します。
4. **OK** をクリックして登録を開始します。
5. TPS がユーザー認証用に設定されている場合は、認証ダイアログにユーザー認証情報を入力して、**Submit** をクリックします。



6. TPS がキーを DRM にアーカイブするように設定されている場合、登録プロセスでキーの生成およびアーカイブが開始されます。

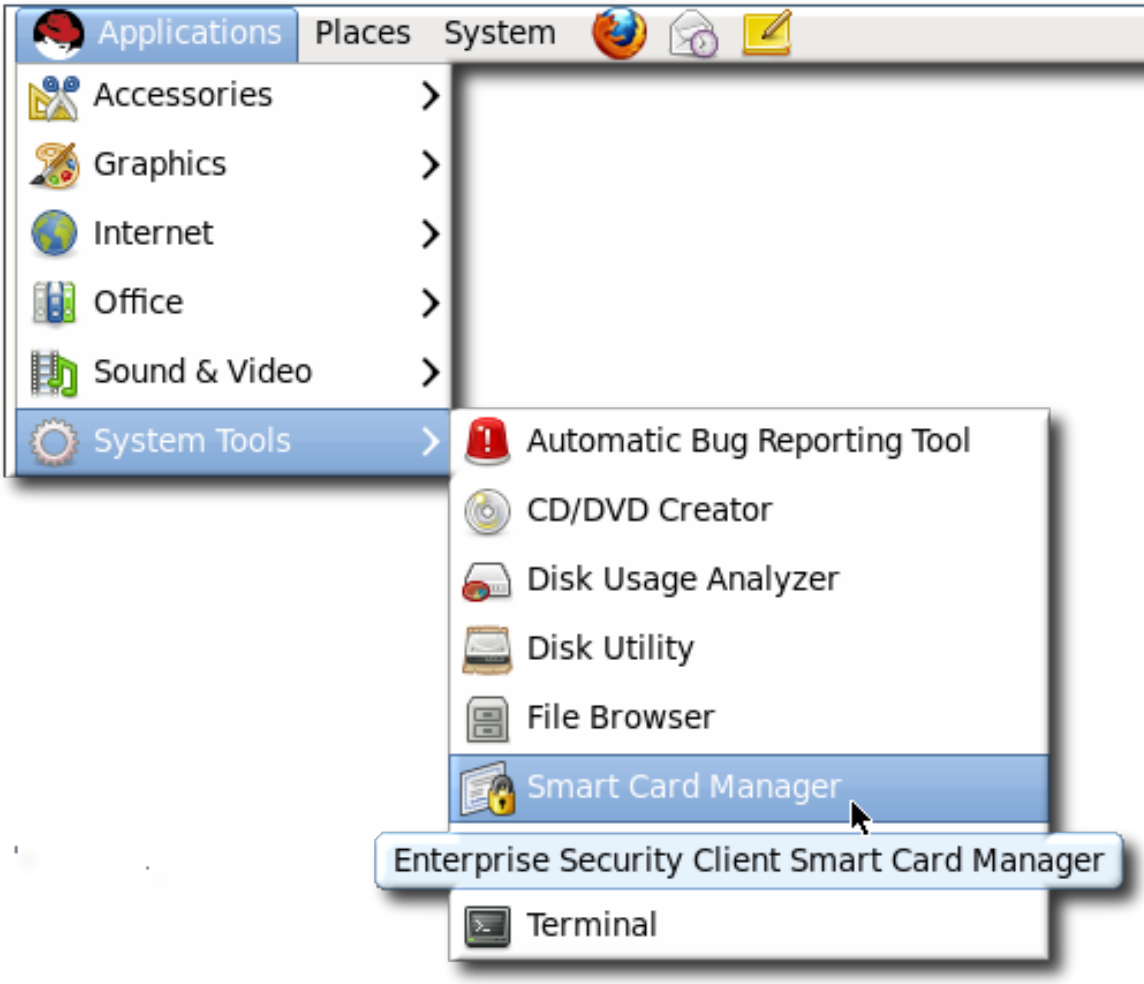
登録が完了すると、スマートカードのステータスが ENROLLED として表示されます。

5.5. 問題の診断

Enterprise Security Client には、基本的な診断ツールと、スマートカードの挿入や削除、カードのパスワードの変更など、エラーや一般的なイベントを記録する簡単なインターフェースが含まれています。診断ツールは、Enterprise Security Client、スマートカード、および TPS 接続の問題について、ユーザーに対して特定して通知できます。

Diagnostics Information ウィンドウを開くには、次のコマンドを実行します。

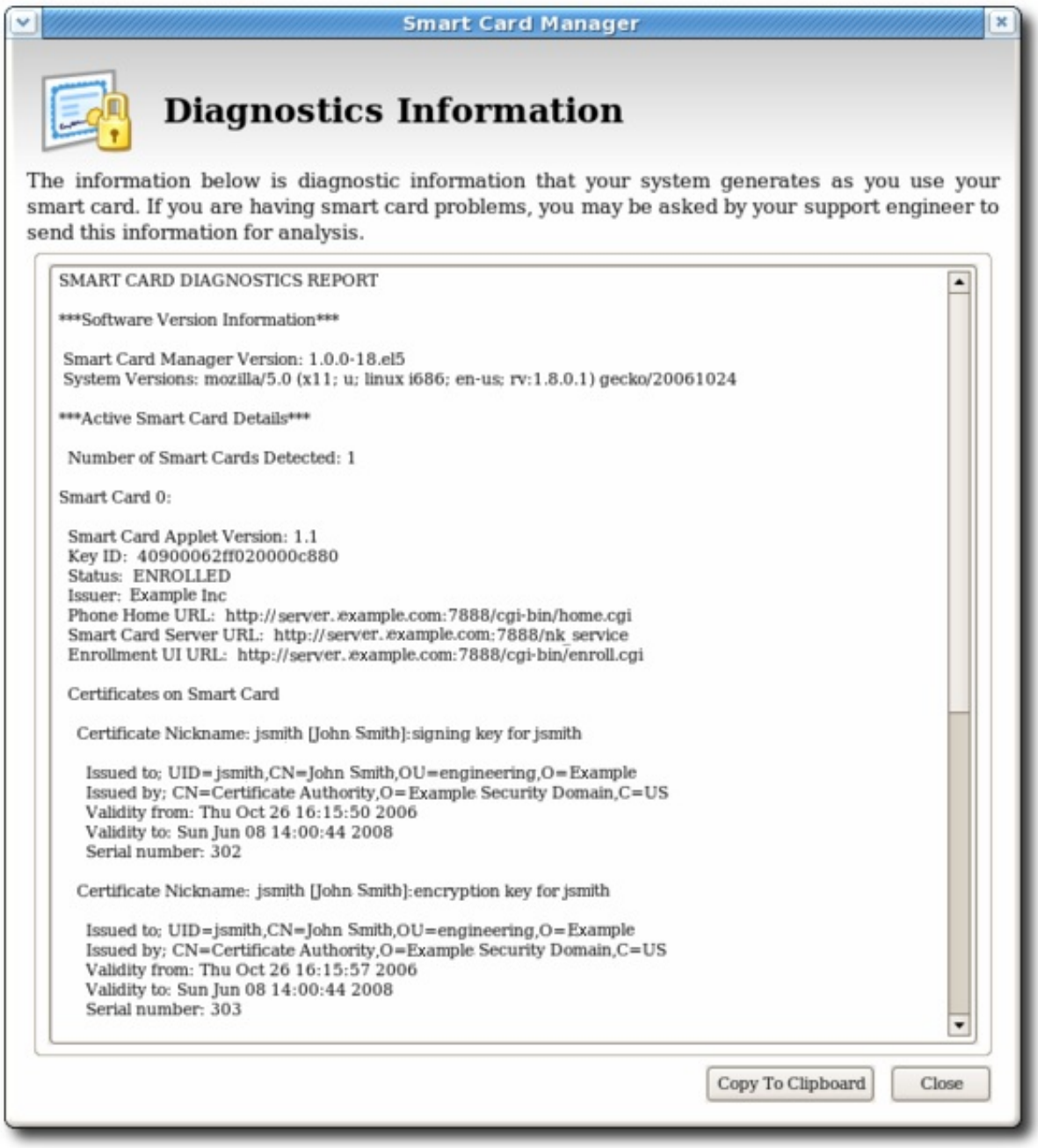
1. Smart Card Manager UI を開きます。



2. 一覧から確認するスマートカードを選択します。

3. **診断** ボタンをクリックします。

4. これにより、選択したスマートカードの **Diagnostic Information** ウィンドウが開きます。



Diagnostics Information 画面には、以下の情報が表示されます。

- Enterprise Security Client のバージョン番号 (スマートカードマネージャーのバージョンとしてリスト)
- クライアントが実行されている XULRunner フレームワークのバージョン情報。
- Enterprise Security Client によって検出されたカードの数。

検出されたカードごとに、以下の情報が表示されます。

- スマートカードで実行しているアプレットのバージョン。
- スマートカードの英数字 ID。
- カードのステータス。以下の 3 つのいずれかになります。

- **NO_APPLET** キーは検出されませんでした。
- **UNINITIALIZED.**キーが検出されても、証明書は登録されていません。
- **ENROLLED.**検出されたカードが、証明書とカード情報に登録されています。

- カードの Phone Home URL。これは、すべての Phone Home 情報の取得元の URL です。
- カード発行者の名前。**Example Corp.** など
- カードの answer-to-reset (ATR) 文字列。これは、スマートカードの異なるクラスの識別に使用できる一意の値です。以下に例を示します。

```
3BEC00FF8131FE45A0000000563333304A330600A1
```

- TPS Phone Home の URL。
- TPS サーバーの URL。これは、Phone HOme 経由で取得されます。
- TPS 登録フォーム URL。これは、Phone HOme 経由で取得されます。
- カードに含まれる各証明書に関する詳細情報。
- 最新の Enterprise Security Client エラーと一般的なイベントの稼働中のログ。

Enterprise Security Client は、2 種類の診断情報を記録します。これは、スマートカードによって返された**エラー**を記録し、Enterprise Security Client 経由で発生した**イベント**を記録します。また、スマートカード設定の基本情報を返します。

5.5.1. エラー

- Enterprise Security Client はカードを認識しません。
- 証明書の登録、パスワードのリセット、フォーマット操作など、スマートカードの操作中に問題が発生します。
- Enterprise Security Client は、スマートカードへの接続を失います。これは、**PCSC** デーモンとの通信で問題が発生した場合に発生する可能性があります。
- Enterprise Security Client と TPS 間の接続が失われます。

スマートカードは、TPS に特定のエラーコードを報告できます。これは、メッセージの原因に応じて TPS **tps-debug.log** または **tps-error.log** ファイルに記録されます。

表5.1 スマートカードのエラーコード

戻りコード	説明
一般的なエラーコード	
6400	特定の診断なし
6700	Lc の誤った長さ

戻りコード	説明
6982	セキュリティーステータスが満たされない
6985	使用条件が満たされない
6a86	正しくない P1 P2
6d00	無効な命令
6e00	無効なクラス
インストール読み込みエラー	
6581	メモリー障害
6a80	データフィールドの誤ったパラメーター
6a84	不十分なメモリー容量
6a88	参照データが見つからない
削除エラー	
6200	アプリケーションを論理的に削除
6581	メモリー障害
6985	参照データを削除できない
6a88	参照データが見つからない
6a82	アプリケーションが見つからない
6a80	コマンドデータの値が正しくない
データ取得エラー	
6a88	参照データが見つからない
ステータス取得エラー	
6310	より多くのデータが利用可能です。
6a88	参照データが見つからない
6a80	コマンドデータの値が正しくない

戻りコード	説明
ロードエラー	
6581	メモリー障害
6a84	不十分なメモリー容量
6a86	正しくない P1/P2
6985	使用条件が満たされない

5.5.2. イベント

- カードを挿入および削除、正常に完了した操作、エラーの原因となるカード操作などのイベントの単純なイベント。
- TPS から Enterprise Security Client にエラーが報告されます。
- NSS 暗号ライブラリーが初期化されています。
- その他の低レベルのスマートカードイベントが検出されています。

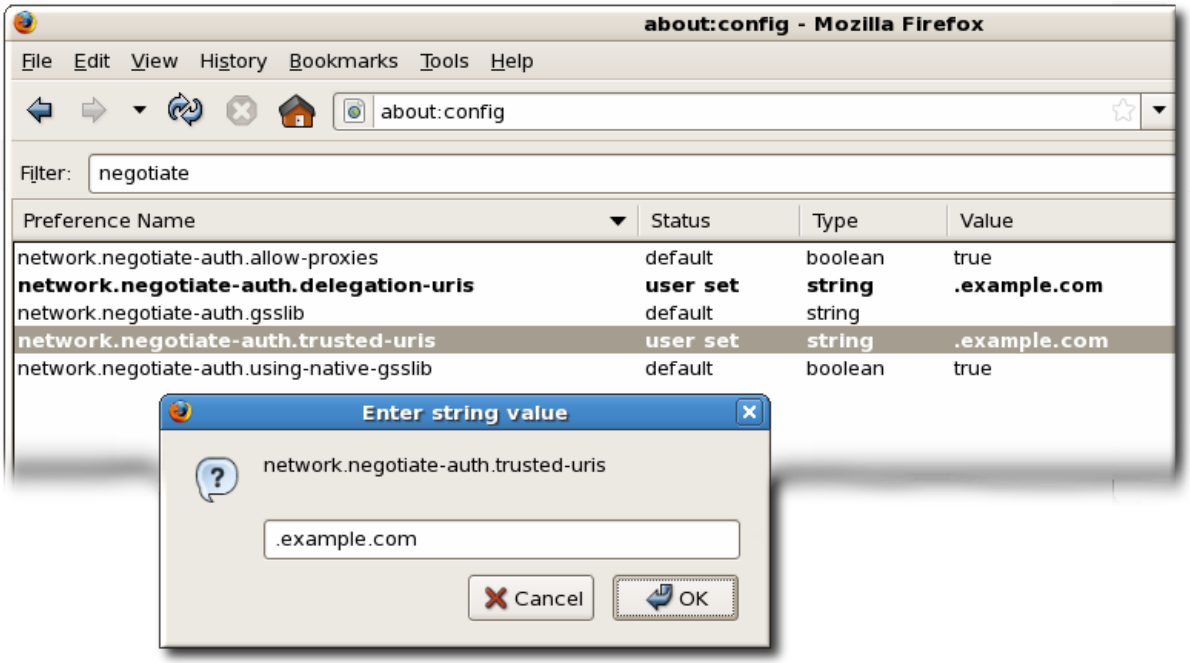
第6章 アプリケーションをシングルサインオン向けに設定

スマートカードは登録後、SSL クライアント認証および S/MIME メールアプリケーションにはスマートカードを使用できます。デフォルトでは、これらのアプリケーションが使用する PKCS #11 モジュールは **/usr/lib/libcoolkeypk11.so** にあります。

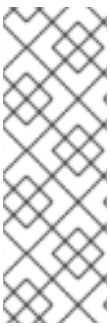
6.1. FIREFOX でシングルサインオンに KERBEROS を使用する設定

Firefox では、Kerberos を使用して、イントラネットサイトや他の保護されている Web サイトへのシングルサインオンを使用できます。Firefox で Kerberos を使用するには、まず Kerberos 認証情報を適切な KDC に送信するように設定する必要があります。

1. Firefox のアドレスバーに、**about:config** と入力して現在の設定オプションの一覧を表示します。
2. **Filter** フィールドに、オプションのリストを制限するために **negotiate** と入力します。
3. **network.negotiate-auth.trusted-uris** エントリーをダブルクリックします。
4. 認証するドメイン名を入力します。



5. 次に、**network.negotiate-auth.trusted-uris** に関しては同じドメインを使用して、**network.negotiate-auth.delegation-uris** エントリーを設定します。



注記

Firefox が Kerberos 認証情報を渡すように設定した後でも、有効な Kerberos チケットが必要になります。Kerberos チケットを生成するには、**kinit** コマンドを実行し、KDC 上のユーザーのユーザーパスワードを指定します。

```
[jsmith@host ~] $ kinit
Password for jsmith@EXAMPLE.COM:
```

Kerberos 認証が機能しない場合は、認証プロセスにおける詳細ロギングをオンにします。

1. Firefox のすべてのインスタンスを閉じます。
2. コマンドプロンプトで、**NSPR_LOG_*** 変数の値をエクスポートします。

```
export NSPR_LOG_MODULES=negotiateauth:5
export NSPR_LOG_FILE=/tmp/moz.log
```

3. **そのシェルから** Firefox を再起動し、Kerberos 認証に失敗していた Web サイトを開きます。
4. メッセージの **nsNegotiateAuth** で、エラーメッセージについて **/tmp/moz.log** ファイルを確認します。

Kerberos 認証では、以下のようなエラーが一般的です。

- 最初のエラーは、認証情報が見つからないことを示しています。

```
-1208550944[90039d0]: entering nsNegotiateAuth::GetNextToken()
-1208550944[90039d0]: gss_init_sec_context() failed: Miscellaneous failure
No credentials cache found
```

これは、Kerberos チケットがないことを意味します (つまり、有効期限が切れたか、生成されていません)。この問題を修正するには、**kinit** を実行して Kerberos チケットを生成し、Web サイトを再度開きます。

- 2 つ目のエラーは、**Kerbros データベースにサーバーが見つかりません** というメッセージが表示されてブラウザーが KDC に接続できない場合です。

```
-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken()
-1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure
Server not found in Kerberos database
```

通常、これは Kerberos 設定の問題です。ドメインを特定するには、正しいエントリーが **/etc/krb5.conf** ファイルの **[domain_realm]** セクションに含まれている必要があります。以下に例を示します。

```
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

- ログにエラーがない場合は、HTTP プロキシーサーバーが Kerberos 認証に必要な HTTP ヘッダーを削除している可能性があります。HTTPS を使用してサイトへの接続を試みます。これにより、要求を変更せずに渡すことができます。

6.2. スマートカードログインの有効化

Red Hat Enterprise Linux サーバーおよびワークステーションにおけるスマートカードでのログインはデフォルトでは有効になっておらず、システム設定で有効にする必要があります。

注記

Red Hat Enterprise Linux にログインする際にシングルサインオンを使用するには、以下のパッケージが必要です。

- nss-tools
- esc
- pam_pkcs11
- coolkey
- ccid
- gdm
- authconfig
- authconfig-gtk
- krb5-libs
- krb5-workstation
- krb5-auth-dialog
- krb5-pkinit-openssl

1. root でシステムにログインします。
2. ネットワーク用のルート CA 証明書をベース 64 形式でダウンロードし、サーバーにインストールします。証明書は、**certutil** コマンドを使用して適切なシステムデータベースにインストールされます。以下に例を示します。

```
# certutil -A -d /etc/pki/nssdb -n "root CA cert" -t "CT,C,C" -i /tmp/ca_cert.crt
```

3. トップメニューで **System** メニューを選択し、**Administration** を選択して、**Authentication**. をクリックします。
4. **高度なオプション** タブを開きます。
5. **Enable Smart Card Support** チェックボックをクリックします。
6. ボタンがアクティブの場合は、**Configure smart card ...** をクリックします。

   スマートカードでは 2 つの動作が設定可能です。

   - **Require smart card for login** チェックボックスにはスマートカードが必要で、基本的にシステムにログインする Kerberos パスワード認証を無効にします。スマートカードを使用して正常にログイン **するまで**は、これは選択しないでください。
   - **カード削除のアクション** は、アクティブなセッション中にスマートカードが削除された場合にシステムが取得する応答を設定します。**Ignore** では、スマートカードが取り除かれるとシステムが通常通り機能し続けます。一方、**Lock** では画面を直ちにロックします。

7. デフォルトでは、証明書が失効したかどうかを確認するメカニズム (オンライン証明書ステータスプロトコル、OCSP、応答) は無効です。有効期限が切れる前に証明書を失効したかどうかを検証するには、***cert_policy*** ディレクティブに **ocsp_on** オプションを追加して OCSP チェックを有効にします。

    1. **pam_pkcs11.conf** ファイルを開きます。

    ```
    vim /etc/pam_pkcs11/pam_pkcs11.conf
    ```

    2. **ocsp_on** オプションが含まれるように、すべての ***cert_policy*** 行を変更します。

    ```
    cert_policy = ca, ocsp_on, signature;
    ```

    注記

    ファイルの解析方法が原因で、**cert_policy** と等号記号の間にスペースが**必要**です。そうでない場合は、パラメーターの解析に失敗します。

8. 「スマートカードの自動登録」 で説明されているように、(個人証明書とキーによる設定で) スマートカードが登録されていない場合、スマートカードを登録します。

9. スマートカードが CAC カードの場合、スマートカードログインに使用される PAM モジュールが特定の CAC カードを認識するように設定する必要があります。

    1. root で、**/etc/pam_pkcs11/cn_map** というファイルを作成ます。

    2. 次のエントリーを **cn_map** ファイルに追加します。

    ```
    MY.CAC_CN.123454 -> login
    ```

    my.CAC_CN.123454 は CAC カードの共通名で、**ログイン** は Red Hat Enterprise Linux ログイン ID です。

注記

スマートカードが挿入されると、**pklogin_finder** ツール (デバッグモード) が最初に、、まずログイン ID をカード上の証明書にマッピングし、証明書の有効性についての情報の出力を試みます。

```
pklogin_finder debug
```

これは、スマートカードを使ってシステムにログインする際の問題を診断する上で役立ちます。

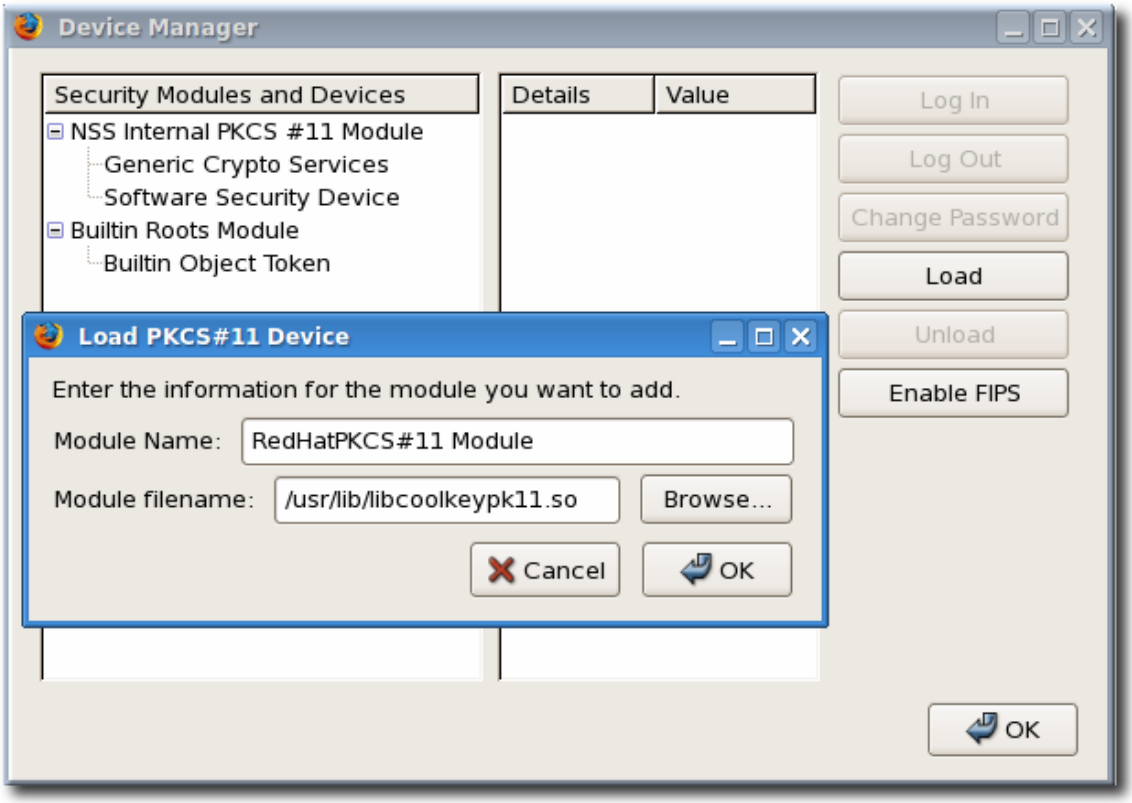
6.3. トークンに対して SSL をサポートするブラウザーの設定

1. Mozilla Firefox で **Edit** メニューを開き、**Preferences** を選択してから **Advanced** をクリックします。

2. **Encryption** タブを開きます。

3. PKCS #11 ドライバーを追加します。

1. **Security Devices** をクリックして **Device Manager** ウィンドウを開き、**Load** ボタンをクリックします。
2. **token key pk11 driver** などのモジュール名を入力します。
3. **Browse** をクリックして、Enterprise Security Client PKCS #11 ドライバーを見つけ、 **OK** をクリックします。デフォルトでは、これらのアプリケーションが使用する PKCS #11 モジュールは **/usr/lib/libcoolkeypk11.so** にあります。



4. CA がまだ信頼されていない場合は、CA 証明書をダウンロードしてインポートします。
   1. CA で **SSL End Entity** ページを開きます。以下に例を示します。

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

   2. **Retrieval** タブをクリックし、**Import CA Certificate Chain** をクリックします。
   3. **Download the CA certificate chain in binary form** をクリックし、**Submit** をクリックします。
   4. 証明書チェーンを保存する適切なディレクトリーを選択し、**OK** をクリックします。
   5. **Edit > Preferences**をクリックして、**Advanced** タブを選択します。
   6. **証明書の表示** ボタンをクリックします。
   7. **Authorities** をクリックし、CA 証明書をインポートします。
5. 証明書信頼関係を設定する手順
   1. **Edit > Preferences**をクリックして、**Advanced** タブを選択します。

2. **証明書の表示** ボタンをクリックします。
3. **Edit** をクリックして、Web サイトへの信頼を設定します。

証明書は SSL に使用できます。

6.4. メールクライアントのトークンでの証明書の使用

1. Mozilla Thunderbird で **Edit** メニューを開き、**Preferences** を選択してから **Advanced** をクリックします。
2. **Certificate** タブを開きます。
3. PKCS #11 ドライバーを追加します。
    1. **Security Devices** をクリックして **Device Manager** ウィンドウを開きます。
    2. **Load** ボタンをクリックします。
    3. **token keypk11 driver** などのモジュール名を入力します。
    4. **Browse** をクリックして、Enterprise Security Client PKCS #11 ドライバーを見つけ、 **OK** をクリックします。デフォルトでは、これらのアプリケーションが使用する PKCS #11 モジュールは **/usr/lib/libcoolkeypk11.so** にあります。
4. CA がまだ信頼されていない場合は、CA 証明書をダウンロードしてインポートします。
    1. CA で **SSL End Entity** ページを開きます。以下に例を示します。

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

    2. **Retrieval** タブをクリックし、**Import CA Certificate Chain** をクリックします。
    3. **Download the CA certificate chain in binary form** をクリックし、**Submit** をクリックします。
    4. 証明書チェーンを保存する適切なディレクトリーを選択し、**OK** をクリックします。
    5. Mozilla Thunderbird で **Edit** メニューを開き、**Preferences** を選択してから **Advanced** をクリックします。
    6. **Certificate** タブを開き、**View Certificates** ボタンをクリックします。
    7. **Authorities** タブをクリックして、CA 証明書をインポートします。
5. 証明書信頼関係を設定する手順
    1. Mozilla Thunderbird で **Edit** メニューを開き、**Preferences** を選択してから **Advanced** をクリックします。
    2. **Certificate** タブを開き、**View Certificates** ボタンをクリックします。
    3. **Authorities** タブで CA を選択し、**Edit** ボタンをクリックします。
    4. Web サイトおよびメールユーザーを特定するために信頼を設定します。

5. **Security** パネルの **Digital Signing** セクションで、**Select** をクリックしてメッセージの署名に使用する証明書を選択します。

6. **Security** パネルの **Encryption** で **Select** をクリックし、メッセージを暗号化および復号化する証明書を選択します。

付録A 改訂履歴

改訂番号はこのマニュアルの編集に関するものであり、Red Hat Enterprise Linux の バージョン番号とは関係ありません。

改訂 6.7-4 Wed Mar 8 2017 Aneta Šteflová Petrová
6.9 GA 公開用バージョン

改訂 6.7-3 Wed May 4 2016 Marc Muehlfeld
6.8 GA 公開用ドキュメントの準備

改訂 6.7-2 Thu Jan 7 2016 Aneta Petrová
更新されたブランドで再構築。

改訂 6.7-1 Tue Jan 5 2016 Aneta Petrová
PAM 設定例のレンダリングを修正しました。

改訂 6.7-0 Tue Jul 14 2015 Tomáš Čapek
6.7 GA リリース向けバージョン

改訂 6.6-1 Fri Dec 19 2014 Tomáš Čapek
スプラッシュページでの分類順序を更新して再構築

改訂 6.6-0 Fri Oct 10 2014 Tomáš Čapek
6.6 GA リリース向けバージョン

改訂 6.4-0 March 28, 2013 Ella Deon Lackey
Publican アップグレードのフォーマットを修正しました。

改訂 6.2-4 December 5, 2011 Ella Deon Lackey
6.2 用のリリース。GA.対応しているスマートカード一覧に PIV カードおよび CAC カードを追加しました。

改訂 6.1-0 Thu May 5, 2011 Ella Deon Lackey
バグの修正。その他の更新。

改訂 6.0-0 Thu Oct 22 2009 Ella Deon Lackey
Red Hat Enterprise Linux 6 の初期ドラフト版