



# SSL サービス セキュア トランザクシ ョンの設定

---

この章では、モジュールの CLI（コマンドライン インターフェイス）から Content Switching Module with SSL（CSM-S）を設定する方法について説明します。

- [PKI の設定（p.8-2）](#)
- [認証局の設定（p.8-46）](#)

## PKI の設定

CSM-S 上の SSL ドータカードは、Secure Socket Layer (SSL) プロトコルを使用し、プライバシー、認証、データの完全性によってデータ セキュア トランザクションを実現します。SSL プロトコルは証明書、公開鍵、および秘密鍵を使用します。

証明書はデジタル ID カードと類似しており、クライアントに対してサーバの、サーバに対してクライアントのアイデンティティを確認します。証明書は認証局が発行し、証明書の発行対象となったエンティティの名前、そのエンティティの公開鍵、証明書の有効期限となる日付を示すタイムスタンプが含まれます。

公開鍵と秘密鍵は、情報を暗号化 / 復号化するために使用する暗号です。公開鍵は制限を伴わずに共有されますが、秘密鍵はいつい共有されません。公開鍵と秘密鍵はペアとして機能します。公開鍵で暗号化されたデータを復号化できるのは、対応する秘密鍵だけです。

各 SSL ドータカードは、最大 256 の SSL クライアントおよびサーバに対する SSL プロキシとして動作します。認証のための証明書を申請するには、各クライアントまたはサーバごとに鍵のペアを設定する必要があります。

証明書は NVRAM (不揮発性 RAM) に保存することを推奨します。モジュールは起動するたびに認証局に問い合わせで証明書を取得したり、自動登録する必要がなくなります。詳細については、「[ルート認証局 \(信頼できるルート\) の設定](#)」(p.8-32) を参照してください。

SSL ドータカードは、SSL サーバとして SSL ドータカードが設定されていて、なおかつクライアントの証明書を認証するサーバプロキシが設定されている場合、または SSL クライアントとして SSL ドータカードが設定されている場合に、外部装置から受け取った証明書を認証します。SSL ドータカードは、受け取った証明書の開始時刻、終了時刻、および署名を検証します。

鍵のペアが壊れていると、有効な証明書が失効していることがあります。失効チェックが必要な場合、SSL ドータカードは認証局から Certificate Revocation List (CRL; 証明書失効リスト) をダウンロードして、受け取った証明書のシリアル番号を検索します。CRL については、「[CRL](#)」(p.8-54) を参照してください。

特定の認証属性値を Access Control List (ACL; アクセス制御リスト) マップと突き合わせることによって、証明書をフィルタリングすることもできます。信頼できる認証局が発行し、認証された証明書だけが受け付けられます。ACL については、「[Certificate Security Attribute-Based Access Control](#)」(p.8-59) を参照してください。



(注)

認証されるのは証明書だけです。証明書の送信元は認証されません。SSL ハンドシェイクの中で、証明書で公開された公開鍵に対応する秘密鍵の所有権について、証明書の送信元が調査されます。この調査で不合格になると、SSL ドータカードが SSL ハンドシェイクを打ち切ります。

SSL ドータカードでは、証明書の送信元が予期した通信セッションのエンドユーザであるか、またはホストであるかを確認することはできません。エンドユーザまたはホストを認証するには、データ フェーズでユーザ名とパスワード、銀行口座番号、クレジットカード番号、または母親の旧姓を使用する、追加の検証手順が必要です。

証明書の送信元が SSL クライアントの場合、SSL ドータカードはデータ フェーズの間にクライアントの証明書から属性を抜き出し、HTTP ヘッダーにそれらの属性を追加できます。このようなヘッダーを受信したサーバシステムはさらに、証明書の件名およびその他の属性を検証し、エンドユーザまたはホストの正統性を判別します。HTTP ヘッダーへの挿入を設定する手順については、「[HTTP ヘッダー挿入](#)」(p.7-14) を参照してください。クライアントの証明書認証を設定する手順については、「[クライアントの証明書の認証](#)」(p.8-47) を参照してください。

次に、Public Key Infrastructure (PKI; 公開鍵インフラストラクチャ) の設定方法について説明します。

- [鍵および証明書の設定 \(p.8-3\)](#)
- [証明書およびトラストポイントの確認 \(p.8-31\)](#)
- [ルート認証局 \(信頼できるルート\) の設定 \(p.8-32\)](#)
- [コンフィギュレーションの保存 \(p.8-33\)](#)
- [鍵および証明書のバックアップ \(p.8-35\)](#)
- [鍵および証明書のモニタおよびメンテナンス \(p.8-35\)](#)
- [プロキシ サービスへの証明書の割り当て \(p.8-37\)](#)
- [証明書の更新 \(p.8-39\)](#)
- [証明書の自動更新および登録の設定 \(p.8-41\)](#)
- [鍵および認証に関する履歴のイネーブル化 \(p.8-42\)](#)
- [ピア証明書のキャッシング \(p.8-43\)](#)
- [証明書の有効期限に関する警告の設定 \(p.8-44\)](#)

## 鍵および証明書の設定

鍵および証明書は、次のいずれかの方法で設定できます。

- Simple Certificate Enrollment Protocol (SCEP) を使用する場合は、次の手順で鍵および証明書を設定します。
  - 鍵のペアを生成します。
  - トラスト ポイントを宣言します。
  - 認証局の証明書を取得します。
  - SSL サーバのために認証局に登録要求を送信します。

詳細については、「[SCEP によるトラストポイントの設定 \(p.8-5\)](#)」を参照してください。

- SCEP を使用しない場合は、次の手順で、手動の証明書登録 (Trivial File Transfer Protocol [TFTP; 簡易ファイル転送プロトコル] およびカット アンド ペースト) 機能を使用して、鍵および証明書を設定します。
  - 鍵のペアを生成またはインポートします。
  - トラスト ポイントを宣言します。
  - 認証局の証明書を取得し、TFTP またはカット アンド ペーストによって PKCS10 ファイルを作成して、トラストポイントを登録します。
  - PKCS10 パッケージを使用し、オフラインで SSL サーバの証明書を要求します。
  - TFTP またはカット アンド ペーストによって、SSL サーバの証明書をインポートします。

詳細については、「[手動証明書登録 \(p.8-11\)](#)」を参照してください。

- 外部の PKI システムを使用する場合は、次の作業を行います。
  - PKCS12 ファイルまたは PEM ファイルを作成します。
  - モジュールにこのファイルをインポートします。

詳細については、「[鍵のペアおよび証明書のインポート / エクスポート \(p.8-22\)](#)」を参照してください。

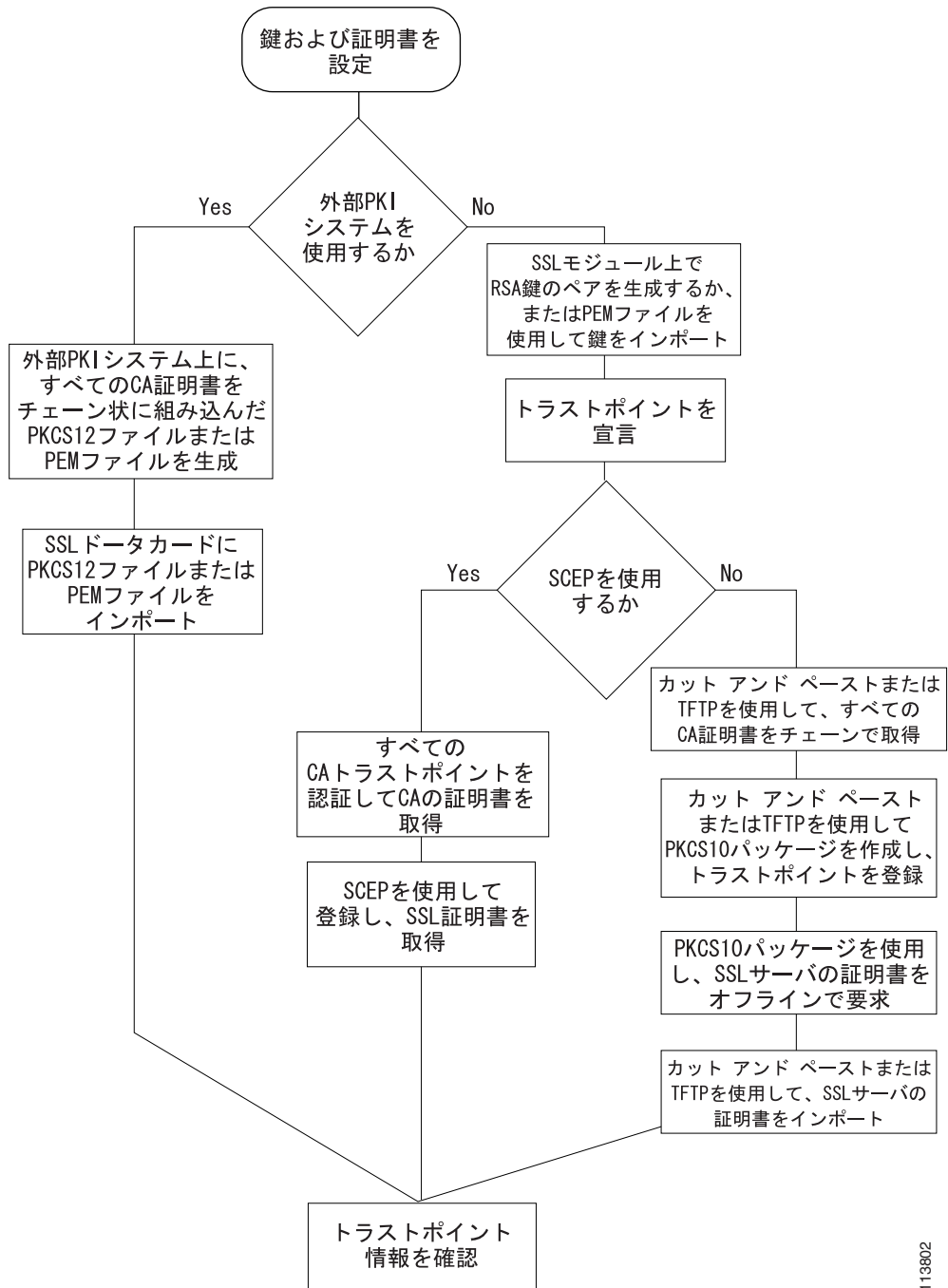
外部 PKI システムは、鍵のペアを生成して、認証局から取得した証明書を登録するサーバまたは PKI 管理システムです。または鍵および証明書のアーカイブシステムです。Public-Key Cryptography Standards (PKCS) は、秘密鍵および証明書を含め、個人のアイデンティティ情報を転送する場合の構文を規定しています。この情報は、暗号ファイルに組み込まれます。暗号ファイルを開くには、パス フレーズが必要です。暗号鍵はこのパス フレーズから引き出されます。



(注) PKCS12 ファイルまたは PEM ファイルをインポートする前に、トラストポイントを設定する必要があります。PKCS12 ファイルまたは PEM ファイルから鍵および証明書をインポートしたときに、トラストポイントがまだなければ自動的に作成されます。

図 8-1 に、鍵および証明書設定の概要を示します。

図 8-1 鍵および証明書設定の概要



113802

## SCEP によるトラストポイントの設定

SCEP を使用してトラストポイントを設定するには、次の作業が必要です。

- [RSA 鍵のペアの生成 \(p.8-5\)](#)
- [トラストポイントの宣言 \(p.8-7\)](#)
- [認証局の証明書取得 \(p.8-8\)](#)
- [証明書の要求 \(p.8-9\)](#)

### RSA 鍵のペアの生成



(注) 最初に生成された鍵のペアによって、モジュール上で Secure Shell (SSH; セキュア シェル) がイネーブルになります。SSH を使用する場合は、SSH に対応する鍵のペアを設定します。「[SSH の設定](#)」(p.7-4) を参照してください。

RSA は、Ron Rivest、Adi Shamir、および Leonard Aldeman が開発した公開鍵暗号システムです。RSA アルゴリズムは、鍵のペアの生成に、認証局および SSL サーバで幅広く使用されています。認証局ごとに、また SSL サーバごとに専用の RSA 鍵のペアを使用します。SSL サーバは証明書を登録するときに、認証局に SSL サーバの公開鍵を送信します。SSL サーバはまた、SSL セッションを確立するときに、証明書を使用してクライアントに SSL サーバのアイデンティティを証明します。

SSL サーバは安全な記憶領域に秘密鍵を保存し、認証局には公開鍵だけを送信します。認証局は自分の秘密鍵を使用して、サーバの公開鍵およびサーバに関するその他の識別情報が含まれている証明書に署名します。

各認証局は秘密鍵を秘密のまま保存し、その秘密鍵を使用して、従属する認証局および SSL サーバの証明書に署名します。認証局には、認証局の公開鍵が指定された証明書があります。

認証局は、1 層または複数層の階層構造で形成されます。最上位の認証局をルート認証局といいます。下位の認証局を中間認証局または従属認証局といいます。ルート認証局は、自己署名入り証明書を使用し、次のレベルの従属認証局の証明書に署名します。その従属認証局は、次に下位レベルの認証局の証明書に署名します。以下同様です。最下位の認証局は、SSL サーバの証明書に署名します。



(注) SSL ドータカードは、最大 8 レベルの認証局（ルート認証局 1 つと最大 7 段階の従属認証局）をサポートします。3 段階（3 層）の登録例については、「[3 層の認証局登録の例](#)」(p.8-10) を参照してください。

このような証明書は、サーバの証明書を最下部、ルート認証局の自己署名入り証明書を最上部に配置したチェーンを形成します。各署名は、発行側認証局の秘密鍵を使用して、証明書本体のハッシュダイジェストを暗号化することによって形成されます。署名が証明書本体の末尾に添付されて、証明書が完成します。

SSL セッションの確立時に、SSL サーバは証明書チェーンをクライアントに送信します。クライアントは、次に上位の証明書から公開鍵を取り出して、証明書本体に添付された署名を復号化することによって、各証明書の署名を順番に検証します。復号結果が証明書本体のハッシュダイジェストと比較されます。チェーンに含まれている認証局の証明書の 1 つが、クライアントの専用データベースに保存されている信頼できる認証局の証明書の 1 つと一致すると、検証が終了します。

チェーンにおける最上位レベルの認証局の証明書に到達し、信頼できる自己署名入り証明書と一致しなかった場合、クライアントはセッションを中断するか、または証明書を表示して信頼できるかどうかの判断をユーザに要求することができます。

サーバを認証した SSL クライアントは、サーバの証明書に含まれている公開鍵を使用して、機密情報を復号化してサーバに送信します。SSL サーバはその秘密鍵を使用して機密情報を復号化します。両者は交換した機密情報と 2 つのランダムな番号を使用し、残りの SSL セッションでデータの暗号化、復号化、および整合性チェックに必要な鍵の素材を生成します。



(注) SSL ドータカードがサポートするのは、汎用鍵だけです。

汎用鍵の生成時には、RSA 鍵のペアが 1 つだけ生成されます。名前付きの鍵のペアを使用して複数の RSA 鍵のペアを用意すると、Cisco IOS ソフトウェアがアイデンティティの証明書ごとに異なる鍵のペアを維持できるようになります。したがって、鍵のペアに名前を指定することを推奨します。



(注) 生成された鍵のペアは、システム メモリ (RAM) に保存されます。電源障害が発生すると、またはモジュールがリセットされると、鍵のペアは失われます。**copy system:running-config nvram:startup-config** コマンドを入力して、モジュール NVRAM 内のプライベート コンフィギュレーション ファイルに実行コンフィギュレーションと鍵のペアを保存する必要があります。

RSA 鍵のペアを生成する手順は、次のとおりです。

コマンド	目的
<code>ssl-proxy(config)# crypto key generate rsa [usage-keys   general-keys] label key-label [exportable<sup>1</sup>] [modulus size]</code>	RSA 鍵のペアを生成します。

1. **exportable** キーワードでは、鍵のエクスポートが可能であることを指定します。鍵がエクスポート可能であることを指定できるのは、鍵の生成時です。いったんエクスポート可能またはエクスポート不可能として生成された鍵をあとから変更することはできません。



(注) RSA 鍵を生成するときには、係数の長さをビット数で指定するように要求されます。SSL ドータカードがサポートする係数の長さは 512、768、1024、1536、および 2048 ビットです。512 または 768 も指定できますが、最小係数長として 1024 を推奨します。係数が長くなるほど生成にも使用にも時間がかかりますが、それだけセキュリティが強化されます。

次に、特殊な用途の RSA 鍵を生成する例を示します。

#### crypto key generate rsa usage-keys

```
The name for the keys will be: myrouter.example.com
Choose the size of the key modulus in the range of 360 to 2048 for your Signature
Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>

Generating RSA keys.... [OK].

Choose the size of the key modulus in the range of 360 to 2048 for your Encryption
Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>

Generating RSA keys.... [OK].
```

次に、汎用の RSA 鍵を生成する例を示します。



(注) 特定用途の鍵と汎用の鍵を両方とも生成することはできません。どちらか一方だけです。

```
ssl-proxy(config)# crypto key generate rsa general-keys label kp1 exportable

The name for the keys will be: kp1
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.
How many bits in the modulus [512]: 1024
Generating RSA keys.... [OK].
```

## トラストポイントの宣言

証明書ごとにモジュールに使用させるトラストポイントを 1 つずつ宣言する必要があります。

モジュールに使用させるトラストポイントを宣言し、そのトラストポイントの特性を指定するには、グローバル コンフィギュレーション モードから始めて次の作業を行います。

	コマンド	目的
ステップ 1	ssl-proxy(config)# <b>crypto ca trustpoint</b> trustpoint-label <sup>1</sup>	モジュールに使用させるトラストポイントを宣言します。このコマンドをイネーブルにすると、ca-trustpoint コンフィギュレーション モードが開始されます。
ステップ 2	ssl-proxy(ca-trustpoint)# <b>rsa</b> keypair key-label	証明書と関連付ける鍵のペアを指定します。
ステップ 3	ssl-proxy(ca-trustpoint)# <b>enrollment</b> [mode ra] [retry [period minutes] [count count]] url url	認証局の登録パラメータを指定します。
ステップ 4	ssl-proxy(ca-trustpoint)# <b>ip-address</b> server_ip_addr	(任意) この証明書を使用するプロキシ サービスの IP アドレスを指定します <sup>2</sup> 。
ステップ 5	ssl-proxy(ca-trustpoint)# <b>crl</b> [best-effort   optional   query host:[port]]	(任意) このトラストポイントに関連付けられた証明書を検証するときに、トラストポイントがどのように証明書失効リストを検索するかを指定します。  CRL については、「 <a href="#">CRL</a> 」(p.8-54) を参照してください。



	コマンド	目的
ステップ 6	<code>ssl-proxy(ca-trustpoint)# subject-name line<sup>3, 4</sup></code>	(任意) プロキシ サービスのホスト名を設定します <sup>5</sup> 。
ステップ 7	<code>ssl-proxy(ca-trustpoint)# password password</code>	(任意) チャレンジ パスワードを設定します。
ステップ 8	<code>ssl-proxy(ca-trustpoint)# exit</code>	ca-trustpoint コンフィギュレーション モードを終了します。

1. *trustpoint-label* は鍵の *key-label* と一致させるべきですが、必須ではありません。
2. 一部の Web ブラウザは、SSL サーバの証明書に指定されている IP アドレスと URL 内の IP アドレスを比較します。2 つの IP アドレスが一致しなかった場合、ブラウザはダイアログを表示し、この証明書を受け入れるか、それとも拒否するかをクライアントに問い合わせます。
3. たとえば、**subject-name CN=server1.domain2.com** の場合、*server1* は URL に含まれている SSL サーバ名です。**subject-name** コマンドでは、Lightweight Directory Access Protocol (LDAP) フォーマットを使用します。
4. 件名に指定する引数にカンマが含まれる場合は、引数を引用符で囲む必要があります。例：O="Cisco, Inc."
5. 一部のブラウザは SSL サーバの証明書に含まれる件名の CN フィールドと URL 内のホスト名を比較します。2 つの名前が一致しなかった場合、ブラウザはダイアログを表示し、この証明書を受け入れるか、それとも拒否するかをクライアントに問い合わせます。また、証明書の CN フィールドが未定義の場合、ブラウザによっては SSL セッションの確立を拒否し、メッセージを出さずに単純にセッションを終了する場合もあります。

次に、トラストポイント PROXY1 を宣言して接続を確認する例を示します。

```
ssl-proxy(config)# crypto ca trustpoint PROXY1
ssl-proxy(ca-trustpoint)# rsakeypair PROXY1
ssl-proxy(ca-trustpoint)# enrollment url http://exampleCA.cisco.com
ssl-proxy(ca-trustpoint)# ip-address 10.0.0.1
ssl-proxy(ca-trustpoint)# password password
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# subject-name C=US; ST=California; L=San Jose; O=Cisco;
OU=Lab;
CN=host1.cisco.com
ssl-proxy(ca-trustpoint)# end
ssl-proxy# ping example.cisco.com
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
ssl-proxy#
```

## 認証局の証明書取得

トラストポイントごとに、認証局の公開鍵が指定された証明書を取得する必要があります。複数のトラストポイントで、同じ認証局を使用できます。



(注) 認証局に接続して証明書の有効なフィンガープリントを入手し、コンソールに表示されたフィンガープリントを確認します。

認証局の公開鍵が指定されている証明書を取得するには、グローバル コンフィギュレーション モードで次の作業を行います。

コマンド	目的
<code>ssl-proxy(config)# crypto ca authenticate trustpoint-label</code>	認証局の公開鍵が指定されている証明書を取得します。トラストポイントの宣言時に入力したのと同じ <i>trustpoint_label</i> を入力します。



認証局の証明書を取得する例を示します。

```
ssl-proxy(config)# crypto ca authenticate PROXY1
Certificate has the following attributes:
Fingerprint: A8D09689 74FB6587 02BFE0DC 2200B38A
% Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.
ssl-proxy(config)# end
ssl-proxy#
```

## 証明書の要求

トラストポイントごとに、認証局から署名入りの証明書を取得する必要があります。

認証局に署名入りの証明書を要求するには、グローバル コンフィギュレーション モードで次の作業を行います。

コマンド	目的
ssl-proxy(config)# <b>crypto ca enroll</b> trustpoint-label <sup>1</sup>	トラストポイントに対応する証明書を要求します。

1. 任意で、コンフィギュレーションに保存されないチャレンジパスワードを作成することもできます。証明書を取り消さなければならない場合、このパスワードが必要なので、パスワードを覚えておいてください。



(注)

**crypto ca enroll** コマンドの入力後、証明書を受け取るまでの間にモジュールまたはスイッチが再起動した場合は、もう一度コマンドを入力し、認証局の管理者に通知する必要があります。

次に、証明書を要求する例を示します。

```
ssl-proxy(config)# crypto ca enroll PROXY1
%
% Start certificate enrollment..

% The subject name in the certificate will be: C=US; ST=California; L=San Jose;
O=Cisco; OU=Lab; CN=host1.cisco.com
% The subject name in the certificate will be: host.cisco.com
% The serial number in the certificate will be: 00000000
% The IP address in the certificate is 10.0.0.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
Fingerprint: 470DE382 65D8156B 0F84C2AF 4538B913

ssl-proxy(config)# end
```

トラストポイントの設定後、「[証明書およびトラストポイントの確認](#)」(p.8-31) を参照して証明書とトラストポイントの情報を確認してください。

### 3 層の認証局登録の例

SSL ドータカードは、最大 8 レベルの認証局（ルート認証局 1 つと最大 7 段階の従属認証局）をサポートします。

次に、3 段階の認証局を設定する例を示します。

#### 鍵の生成

次に、鍵を生成する例を示します。

```
ssl-proxy(onfig)# crypto key generate rsa general-keys label key1 exportable
The name for the keys will be:key1
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

#### トラストポイントの定義

次に、トラストポイントを定義する例を示します。

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.1
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=ste.cisco.com
ssl-proxy(ca-trustpoint)# rsakeypair key1
ssl-proxy(ca-trustpoint)# show
  enrollment url tftp://10.1.1.3
  serial-number
  password 7 02050D480809
  subject-name CN=ste.cisco.com
  rsakeypair key1
end

ssl-proxy(ca-trustpoint)# exit
```

#### 3 つの認証局の認証（1 つのルートおよび 2 つの従属認証局）

```
ssl-proxy(config)# crypto ca authenticate 3tier-root
Certificate has the following attributes:
Fingerprint:84E470A2 38176CB1 AA0476B9 C0B4F478
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate 3tier-sub1
Certificate has the following attributes:
Fingerprint:FE89FB0D BF8450D7 9934C926 6C66708D
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate tp-proxy1
Certificate has the following attributes:
Fingerprint:6E53911B E29AE44C ACE773E7 26A098C3
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
```

### 3 層の認証局への登録

```
ssl-proxy(config)# crypto ca enroll tp-proxy1
%
% Start certificate enrollment ..

% The fully-qualified domain name in the certificate will be:ste.
% The subject name in the certificate will be:ste.
% The serial number in the certificate will be:B0FFF0C2
% Include an IP address in the subject name? [no]:
Request certificate from CA? [yes/no]:yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

ssl-proxy(config)#      Fingerprint: 74390E57 26F89436 6FC52ABE 24E23CD9

ssl-proxy(config)#
*Apr 18 05:10:20.963:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

## 手動証明書登録

手動証明書登録 (TFTP およびカット アンド ペースト) 機能を使用すると、証明書要求を作成したり、認証局の証明書およびルータの証明書を受け付けたりできます。このような作業は、TFTP サーバまたは手動のカット アンド ペースト操作で行います。TFTP または手動のカット アンド ペーストで登録する状況は、次のとおりです。

- 認証局が SCEP をサポートしていない場合 — これが要求と証明書を送受信する最も一般的な方式です。
- ルータと認証局間のネットワーク接続が不可能な場合 — Cisco IOS ソフトウェアが動作しているルータが証明書を取得する方法が規定されます。

手動証明書登録 (TFTP およびカット アンド ペースト) 機能の設定方法については、次の URL を参照してください。

<http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftmancrt.htm>



(注)

CRL サーバが到達不能または CRL ダウンロードパスが存在しなかったことが原因で、CRL をダウンロードできなかった場合、証明書のインポートに失敗する可能性があります。インポートプロセスに結合されているすべてのトラストポイントが CRL をダウンロードできるようにする必要があります。CRL パスが存在しない場合、または CRL サーバが到達不能な場合は、インポートプロセスに結合されているすべてのトラストポイントに **crl optional** コマンドを入力する必要があります。**show crypto ca certificates** コマンドを入力すると、すべての証明書に関する情報が表示されます。認証局の証明書の表示から、関連付けられているトラストポイントのリストが得られます。これらすべてのトラストポイントに **crl optional** コマンドを入力します。

たとえば、3 層の階層型認証局構造 (ルート認証局、従属認証局 1、従属認証局 2) において、従属認証局 1 の証明書をインポートする場合は、ルート認証局に関連付けられたすべてのトラストポイントに **crl optional** コマンドを入力します。同様に、従属認証局 2 の証明書をインポートする場合は、ルート認証局および従属認証局 1 に関連付けられたすべてのトラストポイントに **crl optional** コマンドを入力します。

証明書を正しくインポートしたあとで、トラストポイントに関する元の CRL オプションを復元できます。

## TFTP による証明書登録の設定（1 層の認証局）

TFTP による証明書登録を設定する手順は、次のとおりです。

### ステップ 1 トラストポイントを設定します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto ca trustpoint tftp_example
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2/win2k
ssl-proxy(ca-trustpoint)# rsakeypair pair3
ssl-proxy(ca-trustpoint)# exit
```

### ステップ 2 トラストポイントに対応する証明書を要求します。

```
ssl-proxy(config)# crypto ca enroll tftp_example
% Start certificate enrollment ..

% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
% The subject name in the certificate will be: ssl-proxy.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 00000000
% Include an IP address in the subject name? [no]:
Send Certificate Request to tftp server? [yes/no]: yes
% Certificate request sent to TFTP Server
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
ssl-proxy(config)#      Fingerprint:  D012D925 96F4B5C9 661FEC1E 207786B7
!!
```

### ステップ 3 認証局の公開鍵が指定されている証明書を取得します。

```
ssl-proxy(config)# crypto ca auth tftp_example
Loading win2k.ca from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 1436 bytes]

Certificate has the following attributes:
Fingerprint: 2732ED87 965F8FEB F89788D4 914B877D
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
```

### ステップ 4 サーバの証明書をインポートします。

```
ssl-proxy(config)# crypto ca import tftp_example cert
% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]: yes
% Request to retrieve Certificate queued

ssl-proxy(config)#
Loading win2k.crt from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 2112 bytes]

ssl-proxy(config)#
*Apr 15 12:02:33.535: %CRYPTO-6-CERTRET: Certificate received from Certificate
Authority
ssl-proxy(config)#
```

## カット アンド ペーストによる証明書登録の設定（1 層の認証局）

カット アンド ペーストによる証明書登録を設定する手順は、次のとおりです。

## ステップ 1 RSA 鍵のペアを生成します。

```
ssl-proxy(config)# crypto key generate rsa general-keys label CSR-key exportable
The name for the keys will be:CSR-key
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

## ステップ 2 トラストポイントを設定します。

```
ssl-proxy(config)# crypto ca trustpoint CSR-TP
ssl-proxy(ca-trustpoint)# rsa keypair CSR-key
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=abc, OU=hss, O=cisco
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
```

## ステップ 3 トラストポイントに対応する証明書を要求します。

```
ssl-proxy(config)# crypto ca enroll CSR-TP
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=abc, OU=hss, O=cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:B0FFFF22E
% Include an IP address in the subject name? [no]:no
Display Certificate Request to terminal? [yes/no]:yes
Certificate Request follows:

MIIBWjCCASsCAQAwYTEOMAwGA1UEChMFY2l2Y28xDDAKBgNVBAsTA2hzc2EMMAoG
A1UEAxMDYWJjMTMwDwYDVQQFEWhCMEZGRjIyRTAgBgkqhkiG9w0BCQIWE3NzbC1w
cm94eS5jaXNjb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt
301BVVK1qAE/agsuzIaa15YZft3bDb9t3pPncKh0ivBTgVKpJiLPWGPZPjdbtejxQ
tYSF77R1pmhK0WSKPuu7fJPYr/Cho800UzkRAgMBAAGgITAfBgkqhkiG9w0BCQ4x
EjAQMA4GA1UdDwEB/wQEAwIFoDANBgkqhkiG9w0BAQQFAAOBgQC2GIX06/hihXHA
DA5sOpXgLS01rMP8PF4bZDdlpWLVBSOrp4S1L7hH9P2NY9rgZAJhDTRfGGm179JY
G0tUuCyPYPkpb0S5VGTUrHvvUWekleKq2d91kfgbkRmJmHBaB2Ev5DNBcV11SIMX
RULG7oUafU6sxnDWqbMseToF4WrLPg==

---End - This line not part of the certificate request---
```

Redisplay enrollment request? [yes/no]:**no**

**ステップ 4** 認証局の証明書をインポートします。

```
ssl-proxy(config)# crypto ca authenticate CSR-TP

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIICxzCCAjCgAwIBAgIBADANBgkqhkiG9w0BAQQFADBSMQswCQYDVQQGEwJBVTET
MBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJuZXQvV2lkZ2l0cyBQ
dHkgTHRkMQswCQYDVQQDEwJjYTAeFw0wMzA2MjYyMjM4MDlaFw0wODEyMTYyMjM4
MDlaMFIXCzAJBgNVBAYTAkFVMRMwEQYDVQQIEWpTb211LVN0YXRlMSEwHwYDVQQK
ExhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQxCzAJBgNVBAMTAmNhMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCcG9ObqOLmf0cAskF48jz8X7ZQxT1H68OQKNC3
ks95vkGbOAA/1/R4ACQ3s9iPkcGQVqi4Dv8/iNG/1mQo8HBwtR9VgG0l8IGBbuiZ
dlarYnHUz6Bm/HzE1RXVOY/VmyPOVevYy8/cYhw/xOE9BYQOyP15Chi8nhIS5F
+WWoHQIDAQABO4GsMIGpMB0GA1UdDgQWBBS4Y+/LSXKDrw5N5m/tgCzu/W81PDB6
BgNVHSMeczBxgBS4Y+/LSXKDrw5N5m/tgCzu/W81PKFWpFQwUjELMAkGA1UEBhMC
QVUxEzARBgNVBAgTC1NvbWUuU3RhdGUxITAFBgNVBAoTGEIudGVybmV0IFdpZGdp
dHMgUHR5IEEx0ZDElMAkGA1UEAxMCY2GCAQAwdAYDVR0TBAAUwAwEB/zANBgkqhkiG
9w0BAQQFAAOBqQB/rPdLFVuycbaJQuCdFQG7kl/XBNI7aY3IL3Lkeumt/nXD+eCn
RpYE5WWY8X1Aizqnj4bqFdqPqYdD7Lg8viwqm2tQmU6zCsdaKhL1J7FCWbfs2+Z5
oNV2Vsqx0Ftnf8en/+HtyS2AdXHreThfgkXz3euXD0ISMfVKRy81o4EdzA==
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:B8B35B00 095573D0 D3B8FA03 B6CA8934
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)#
```

**ステップ 5** サーバの証明書(ステップ 4 で証明書をインポートした認証局によって発行されたサーバの証明書)をインポートします。

```
ssl-proxy(config)# crypto ca import CSR-TP certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com

Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIB7TCCA0YCAQAwDQYJKoZIhvcNAQEEBQAwUjELMAkGA1UEBhMCQVUxEzARBgNV
BAGTC1NvbWUuU3RhdGUxITAFBgNVBAoTGEIudGVybmV0IFdpZGdpdHMgUHR5IEEx0
ZDElMAkGA1UEAxMCY2EwHhcNMjM0MTIwMDAxMzE2WhcNMjM0MTIwMDAxMzE2WjAs
MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMQwwCgYDVQQDEwNhYmMwZ8w
DQYJKoZIhvcNAQEEBQADgY0AMIGJAoGBALt7O6tt30LBVVK1qAE/agsuzIaa15YZ
ft3bDb9t3pPncKh0ivBTgVKpJiLPWGPjdbtejxQksuSY589V+GMDrO9B4Sxn+5N
p2bQmd745NvI4gorNRvXcdjme+/Sze+bBSBcKAwNtYSF77R1pmhK0WSKPuu7fJPY
r/Cbo80OUzkRAGMBAAEWdQYJKoZIhvcNAQEEBQADgYEAjqJ9378P6Gz69Ykplw06
Powp+2rbe2iFBrElxE09BL6G6vzcBQgb5W4uwqxe7SIHrHsS0/7Be3zeJn10seWx
/KVj7I02iPgrwUa9DLavwrTyaa0KtTpti/i5nIwTNh5xkp2bBJQikD4TEK7HAvXf
HQ9SyB3YZJk/Bjp6/eFHEfU=
-----END CERTIFICATE-----

% Router Certificate successfully imported

ssl-proxy(config)#^Z
```

## TFTP による証明書登録の設定（3 層の認証局）

TFTP による証明書登録を設定する手順は、次のとおりです。

**ステップ 1** RSA 鍵のペアを生成します。

```
ssl-proxy(config)# crypto key generate rsa general-keys label test-3tier exportable
The name for the keys will be:test-3tier
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

**ステップ 2** トラストポイントを設定します。

```
ssl-proxy(config)# crypto ca trustpoint test-3tier
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=test-3tier, OU=hss, O=Cisco
ssl-proxy(ca-trustpoint)# rsa keypair test-3tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-3tier
ssl-proxy(ca-trustpoint)# exit
```

**ステップ 3** Certificate Signing Request (CSR) を生成して TFTP サーバに送信します。

```
ssl-proxy(config)# crypto ca enroll test-3tier
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test-3tier, OU=hss, O=Cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:B0FFF22E
% Include an IP address in the subject name? [no]:
Send Certificate Request to tftp server? [yes/no]:yes
% Certificate request sent to TFTP Server
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

ssl-proxy(config)# Fingerprint: 19B07392 319B2ACF F8FABE5C 52798971

ssl-proxy(config)#
!!
```

**ステップ 4** CSR を使用して 3 層の認証局からオフラインで SSL 証明書を取得します。



**ステップ 5** 3つの認証局（1つのルートおよび2つの従属認証局）を認証します。

```

ssl-proxy(config)# crypto ca trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-1tier
Loading test-1tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1046 bytes]

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.

ssl-proxy(config)# crypto ca trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-2tier
Loading test-2tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1554 bytes]

Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

ssl-proxy(config)# crypto ca authenticate test-3tier
Loading test-3tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1545 bytes]

Certificate has the following attributes:
Fingerprint:2F2E44AC 609644FA 5B4B6B26 FDBFE569
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

```

**ステップ 6** サーバの証明書をインポートします。

```

ssl-proxy(config)# crypto ca import test-3tier certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]:yes
% Request to retrieve Certificate queued

ssl-proxy(config)#
Loading test-3tier.crt from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1608 bytes]

ssl-proxy(config)#
*Nov 25 21:52:36.299:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
ssl-proxy(config)# ^Z

```

## カット アンド ペーストによる証明書登録の設定（3 層の認証局）

カット アンド ペーストによる証明書登録を設定する手順は、次のとおりです。

### ステップ 1 RSA 鍵のペアを生成します。

```
ssl-proxy(config)# crypto key generate rsa general-keys label tp-proxy1 exportable
The name for the keys will be:tp-proxy1
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

### ステップ 2 トラストポイントを設定します。

```
ssl-proxy(config)# crypto ca trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment ter
ssl-proxy(ca-trustpoint)# rsakeypair tp-proxy1
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=test
ssl-proxy(ca-trustpoint)# exit
```

### ステップ 3 トラストポイントに対応する証明書を要求します。

```
ssl-proxy(config)# crypto ca enroll tp-proxy1
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test
% The fully-qualified domain name in the certificate will be:ssl-proxy.
% The subject name in the certificate will be:ssl-proxy.
% The serial number in the certificate will be:B0FFF14D
% Include an IP address in the subject name? [no]:no
Display Certificate Request to terminal? [yes/no]:yes
Certificate Request follows:

MIIBnDCCAQUCAQAwOzENMAAGA1UEAxMEdGVzdDEqMA8GA1UEBRMIQjBGRkYxNEQw
FwYJKoZIhvcNAQkCFgpzc2wtcHJveHkuMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQDFx1ol9IXoAx4fyUhaXH6s4p5t9soIZ1gvLtVX6Fp6zfuX47os5TGJH/IX
zV9B4e5Kv+w1MD0AvTh+/tvyAP3TMpCdpHYosd2VaTIgExpHf4M5Ruh8IebVKV25
rraIpNiS0PvPLFCrw4UfJVNpsc2XBxBhpT+FS9y67Lq1hfsN4wIDAQABoCEwHwYJ
KoZIhvcNAQkOMRIwEDAOBgNVHQ8BAf8EBAMCBAAwDQYJKoZIhvcNAQEEBQADgYEA
kOIjd1KNJdKLMf33YELRd3MW/ujJIuiTlJ8RYVbw1eE8JQf68TTdKiYqzQcoMgsp
ez3vSPxXFZ/c6naXdVyrTikTX3GZ1mu+UOvV6/Jaf5QcXa9tAi3fgyguV7jQMPjk
Qj2GrwXjcqZGOMBh6Kq6s5UPsIDgrL036I42B6B3EQ=

---End - This line not part of the certificate request---

Redisplay enrollment request? [yes/no]:no
```

### ステップ 4 （ステップ 3 から）3 層の認証局が署名した証明書要求を取得します。

**ステップ 5** すべての認証局（1 つのルートおよび 2 つの従属認証局）を定義してインポートします。

- a. ルート認証局と従属 1 認証局に対応する 2 つのトラストポイントを定義します。



(注) **tp-proxy1** を使用して、従属 2 認証局の証明書をインポートします。

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
ssl-proxy(ca-trustpoint)# exit
```

- b. ルート認証局の証明書をインポートします。

```
ssl-proxy(config)# crypto ca authenticate 3tier-root
```

```
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
```

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAOGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcml5TERMA8GA1UEBxMic2Fu
IGpvc2UxZjAMBgNVBAoTBWNPc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTExNDgwMloXDTEzMTEwMTEx
NTczOVowdTElMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEwETAPBgNV
BACgTCHNhbG1mb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECjMDaHh0aHh0aHh0
VQDEdDzaW1wc29uLWRLdnRlc3Qtcml5dGV2dGV2dGV2dGV2dGV2dGV2dGV2dGV2dGV2
MEgCQQCWEibAnU1VqQUNUn0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVzR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVzR0OBByEFcyGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wGywwQ6BBOD+GPWh0dHA6Ly9jaXNjb3NlMQ4wDAYDVQQKEwVjaXNjbz
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcml5dGV2dGV2dGV2dGV2dGV2dGV2dGV2dGV2dGV2
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBMB4wDQYJKoZIhvcNAQEFBQADANBgkqhkiG9w0BAQUFAANBACBqelwy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
```

```
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

```
ssl-proxy(config)# crypto ca authenticate 3tier-sub1
```

-----BEGIN CERTIFICATE-----

```
Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
```



- ```
ssl-proxy (config) #^Z
```

## 鍵のペアおよび証明書のインポート / エクスポート

PKCS12 ファイル形式または Privacy-Enhanced Mail (PEM) ファイル形式を使用することによって、鍵のペアと証明書をインポートできます。

ここでは、鍵のペアと証明書をインポートまたはエクスポートする方法について説明します。

- [PKCS12 ファイルのインポート / エクスポート \(p.8-22\)](#)
- [PEM ファイルのインポート / エクスポート \(p.8-24\)](#)



(注) モジュール上の SSL ソフトウェアにテスト用の PKCS12 ファイル (test/testssl.p12) が組み込まれています。テスト目的で、または概念を確認する目的で、このファイルを NVRAM にインストールできます。PKCS12 ファイルをインストールしたあとで、トラストポイントにインポートし、さらにテスト用に設定したプロキシ サービスに割り当てることができます。テスト用 PKCS12 ファイルのインストール方法については、「[鍵のペアおよび証明書のインポート / エクスポート \(p.8-22\)](#)」を参照してください。



(注) CRL サーバが到達不能または CRL ダウンロードパスが存在しなかったことが原因で、CRL をダウンロードできなかった場合、証明書のインポートに失敗する可能性があります。インポートプロセスに結合されているすべてのトラストポイントが CRL をダウンロードできるようにする必要があります。CRL パスが存在しない場合、または CRL サーバが到達不能な場合は、インポートプロセスに結合されているすべてのトラストポイントに **crl optional** コマンドを入力する必要があります。**show crypto ca certificates** コマンドを入力すると、すべての証明書に関する情報が表示されます。認証局の証明書の表示から、関連付けられているトラストポイントのリストが得られます。これらすべてのトラストポイントに **crl optional** コマンドを入力します。

たとえば、3 層の階層型認証局構造（ルート認証局、従属認証局 1、従属認証局 2）において、従属認証局 1 の証明書をインポートする場合は、ルート認証局に関連付けられたすべてのトラストポイントに **crl optional** コマンドを入力します。同様に、従属認証局 2 の証明書をインポートする場合は、ルート認証局および従属認証局 1 に関連付けられたすべてのトラストポイントに **crl optional** コマンドを入力します。

証明書を正しくインポートしたあとで、トラストポイントに関する元の CRL オプションを復元できます。

## PKCS12 ファイルのインポート / エクスポート

外部 PKI システムを使用すると、PKCS12 にファイルを作成してモジュールにインポートできます。



(注) PKCS12 ファイルを作成する場合は、サーバの証明書からルート証明書までの証明書チェーン全体、公開鍵、および秘密鍵を含めます。モジュールから PKCS12 ファイルを生成してエクスポートすることもできます。



(注) インポートした鍵のペアをエクスポートすることはできません。





(注) SSH を使用する場合は、PKCS12 ファイルのインポート/エクスポート時に、SCP (セキュア ファイル転送) を使用することを推奨します。SCP はホストを認証し、転送セッションを暗号化します。

PKCS12 ファイルのインポート/エクスポート手順は、次のとおりです。

| コマンド                                                                                                                                                                          | 目的                                                                                                                                                        |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ssl-proxy(config)# crypto ca {import   export} trustpoint_label pkcs12 {scp:   ftp:   nvram:   rcp:   tftp:} [pkcs12_filename<sup>1</sup>] pass_phrase<sup>2</sup></pre> | <p>PKCS12 ファイルをインポートまたはエクスポートします。</p> <p> (注) PKCS12 ファイルをインポートする前に、トラストポイントを設定する必要はありません。PKCS12 ファイルから鍵と証明書をインポートしたときに、トラストポイントがまだなかった場合は自動的に作成されます。</p> |

1. *pkcs12\_filename* 値を指定しなかった場合、デフォルトのファイル名 (*trustpoint\_label* の値) を受け入れるか、またはファイル名を入力するように求められます。**ftp:** または **tftp:** の場合、*pkcs12\_filename* の値にフルパスを指定します。
2. パス フレーズを誤って入力すると、エラーになります。

次に、SCP を使用して PKCS12 ファイルをインポートする例を示します。

```
ssl-proxy(config)# crypto ca import TP2 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Source username [ssl-proxy]? admin-1
Source filename [TP2]? /users/admin-1/pkcs12/TP2.p12

Password:password
Sending file modes:C0644 4379 TP2.p12
!
ssl-proxy(config)#
*Aug 22 12:30:00.531:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
ssl-proxy(config)#
```

次に、SCP を使用して PKCS12 ファイルをエクスポートする例を示します。

```
ssl-proxy(config)# crypto ca export TP1 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination username [ssl-proxy]? admin-1
Destination filename [TP1]? TP1.p12

Password:

Writing TP1.p12 Writing pkcs12 file to scp://admin-1@10.1.1.1/TP1.p12

Password:
!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

次に、FTP(ファイル転送プロトコル)を使用して PKCS12 ファイルをインポートする例を示します。

```
ssl-proxy(config)# crypto ca import TP2 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Source filename [TP2]? /admin-1/pkcs12/PK-1024
Loading /admin-1/pkcs12/PK-1024 !
[OK - 4339/4096 bytes]
ssl-proxy(config)#
```

次に、FTP を使用して PKCS12 ファイルをエクスポートする例を示します。

```
ssl-proxy(config)# crypto ca export TP1 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination filename [TP1]? /admin-1/pkcs12/PK-1024
Writing pkcs12 file to ftp://10.1.1.1/admin-1/pkcs12/PK-1024

Writing /admin-1/pkcs12/PK-1024 !!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

PKCS12 ファイルのインポート後、「[証明書およびトラストポイントの確認](#)」(p.8-31) を参照して証明書とトラストポイントの情報を確認してください。

## PEM ファイルのインポート / エクスポート



(注) **crypto ca import pem** コマンドによってインポートされるのは、秘密鍵 (.prv)、サーバの証明書 (.crt)、および発行認証局の証明書 (.ca) だけです。証明書のチェーンに複数レベルの認証局が含まれている場合、ルートと従属認証局の証明書をインポートしてから、このコマンドを実行して認証する必要があります。ルートと従属認証局の証明書をインポートするには、カット アンド ペーストまたは TFTP を使用します。





(注) インポートした鍵のペアをエクスポートすることはできません。



(注) SSH を使用する場合は、PEM ファイルのインポート / エクスポート時に、セキュア ファイル転送 (SCP) を使用することを推奨します。SCP はホストを認証し、転送セッションを暗号化します。

PEM ファイルをインポートまたはエクスポートするには、次のどちらか一方の作業を行います。

| コマンド                                                                                                                                                                       | 目的                                                                                                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ssl-proxy(config)# crypto ca import trustpoint_label pem [exportable] {terminal   url {scp:  ftp:  nvram:  rcp:  tftp:}   usage-keys} pass_phrase<sup>1,2</sup></pre> | <p>PEM ファイルをインポートします。</p> <p> (注) PEM ファイルをインポートする前に、トラストポイントを設定する必要はありません。PEM ファイルから鍵と証明書をインポートしたときに、トラストポイントがまだなかった場合は自動的に作成されます。</p> |
| <pre>ssl-proxy(config)# crypto ca export trustpoint_label pem {terminal   url {scp:  ftp:  nvram:  rcp:  tftp:} [des   3des] pass_phrase<sup>1,2</sup></pre>               | <p>PEM ファイルをエクスポートします。</p> <p> (注) エクスポートされるのは、鍵、サーバの証明書、およびサーバの証明書を発行した認証局だけです。それより上位の認証局は、TFTP のカット アンドペーストでエクスポートする必要があります。</p>      |

1. パス フレーズを誤って入力すると、エラーになります。
2. パス フレーズは、秘密鍵の含まれている PEM ファイルを保護します。PEM ファイルは DES または 3DES で暗号化されます。暗号鍵はこのパス フレーズから引き出されます。証明書が含まれている PEM ファイルは暗号化されません。パス フレーズによる保護也没有ありません。

次に、TFTP を使用して PEM ファイルをインポートする例を示します。



(注) サーバ上に TP5.ca、TP5.prv、および TP5.crt ファイルが存在していなければなりません。

```
ssl-proxy(config)# crypto ca import TP5 pem url tftp://10.1.1.1/TP5 password
% Importing CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.ca]?
Reading file from tftp://10.1.1.1/TP5.ca
Loading TP5.ca from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1976 bytes]

% Importing private key PEM file...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.prv]?
Reading file from tftp://10.1.1.1/TP5.prv
Loading TP5.prv from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 963 bytes]

% Importing certificate PEM file...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.crt]?
Reading file from tftp://10.1.1.1/TP5.crt
Loading TP5.crt from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1692 bytes]
% PEM files import succeeded.
ssl-proxy(config)#end
ssl-proxy#
*Apr 11 15:11:29.901: %SYS-5-CONFIG_I: Configured from console by console
```

次に、TFTP を使用して PEM ファイルをエクスポートする例を示します。

```
ssl-proxy(config)# crypto ca export TP5 pem url tftp://10.1.1.1/tp99 3des password
% Exporting CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.ca]?
% File 'tp99.ca' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.ca!
% Key name: key1
Usage: General Purpose Key
% Exporting private key...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.prv]?
% File 'tp99.prv' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.prv!
% Exporting router certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.crt]?
% File 'tp99.crt' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.crt!
ssl-proxy(config)#
```

PEM ファイルのインポート後、「[証明書およびトラストポイントの確認](#)」(p.8-31)を参照して証明書とトラストポイントの情報を確認してください。

## PEM ファイルのインポート（3 層の認証局の場合）

ここでは、オフライン登録のカット アンド ペースト オプションを使用して、ルート認証局の証明書（第 1 層）と中間認証局の証明書（第 2 層）を取得します。さらに PEM ファイルをインポートすることによって、中間認証局の証明書（第 3 層）、秘密鍵、およびルータの証明書を取得します。

認証局が 3 層構造の場合に PEM ファイルをインポートする手順は、次のとおりです。

### ステップ 1 カット アンド ペーストでルート認証局の第 1 層証明書を取得します。

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate 3tier-root

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIC1zCCAOGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcmlpYTERMA8GA1UEBxMlY2Fu
IGpvc2UxZDjAMBgNVBAoTBWVnc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzB24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTExNDgwMl0XDTEzMTEwMTEx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbg1mb3JuaWEwETAPBgNV
BACgTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdAHNzMSAwHgYD
VQDEExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNU0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFYCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBOD+GPWh0dHA6Ly9jaXNjb3NlMQ4wDAYDVQQKEwVjaXNjbz
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQSB5cmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xkZDZlX0RlY290LWw4ajZvaHBuc1xkZDZlX0RlY290
LUNBMB4XDTEzMTEwMTExNDgwMl0XDTEzMTEwMTExNTczOVowdTELMAkGA1UEBhMCVVM
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

**ステップ 2** カット アンド ペースト で従属認証局 1 の証明書を取得します。

```
ssl-proxy(config)# crypto ca trustpoint 3tier-subca1
ssl-proxy(ca-trustpoint)# enroll terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate 3tier-subca1
```

Enter the base 64 encoded CA certificate.  
End with a blank line or the word "quit" on a line by itself

```
-----BEGIN CERTIFICATE-----
MIETzCCA/mgAwIBAgIKGj0cBwAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMY2FsaWZvcml5TERMA8GA1UEBxMlc2FuIGpvc2Ux
DjAMBGNVBAOTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTEzMzIyMDQyMVoXDTA0MTEzMzIyMTQyMVo
dTELMARGA1UEBhMCMVVMxZARBgNVBAgTCmNhbmG1mb3JuaWEwETAPBgNVBACtCHNh
biBqb3N1MQ4wDAYDVQQKEWVjaXNjbzEMMAoGA1UECjMDAHRzMSAwHgYDVQQDExdz
aW1wc29uLWRLdnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDc
vV48nC2uukoSyGJ/GymCIEXXZMSzpbkYS7eWPaZYyiJDhCIKuUsMgFDRNfMQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAGHMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwSBpjCB04AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTALVTMRMwEQYDVQQIEWpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2l2Y28xDDAKBgNVBAsTA2hzc2EgMB4GA1UE
AxMXc2l2cHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcG
A1UdHwSBjzCBjDBDQEGGp4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuc19DZXJ0RW5y
b2xsL3NpbXBzb24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2l2Y28tbDhgNm9ocG5yXEN1cnRFbnJvbGxccc2ltcHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0EuY3JsmIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
c2NvLWw4ajZvaHBuc19DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBuc19zaW1wc29u
LWRLdnRlc3Qtcml9vdC1DQ85jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuc19DZXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBuc19zaW1wc29uLWRL
dnRlc3Qtcml9vdC1DQ85jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9
ER36ZkdJNlW93gNt2MkpcA07RmcrHln6q5RJ9WbvTxFnONdgpsag1EcOwn97XErH
Z2ow
-----END CERTIFICATE-----
```

```
Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

## Catalyst 6500 シリーズ スイッチ CSM-S インストレーション コンフィギュレーション ノート



```

aXNjbzEMMAoGA1UECXMdAHNzMSAwHgYDVQDEdzaW1wc29uLWRLdnRlc3Qtc3Vi
MS1jYYIKHyiFxAaaaaaABjAoBgNVHREBAf8EHjAcghpzaW1wc29uLTY1MDktc3Rl
LmNpc2NvLmNvbTCBlwYDVROfBIGPMIGMMEOGQaA/hj1odHRwOi8vY2l2Y28tb2pt
eGNuY3p2L0NlcnRFbnJvbGwvc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3JsMEWg
Q6BBhj9maWxlOi8vXFXjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxxaW1wc29u
LWRLdnRlc3Qtc3ViMi1jYS5jcmwwgGCGCCsGAQUFBwEBBIG7MIG4MFkGCCsGAQUF
BzAChklodHRwOi8vY2l2Y28tb2pteGNuY3p2L0NlcnRFbnJvbGwvY2l2Y28tb2pt
eGNuY3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDBbBggrBgEFBQcwAoZP
ZmlsZTovL1xcY2l2Y28tb2pteGNuY3p2XENlcnRFbnJvbGxcY2l2Y28tb2pteGNu
Y3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDANBgkqhkiG9w0BAQUFAANB
ABFh7XeLwvfBtjAR+e5OaUH5KTGJDbeJppOmMFxnFakpgWop9Qg4cHRCQq7V0pAW
iA6VtJOmpYgEIVNTAzAAHR4=
-----END CERTIFICATE-----

% PEM files import succeeded.
ssl-proxy(config)# ^Z
ssl-proxy#
*Dec 4 18:11:49.850:%SYS-5-CONFIG_I:Configured from console by console
ssl-proxy#

```

#### ステップ 4 証明書情報を表示します (任意)。

```

ssl-proxy# show crypto ca certificates tp-proxy1
Certificate
  Status:Available
  Certificate Serial Number:04A0147B000000000010E
  Certificate Usage:General Purpose
  Issuer:
    CN = sub3ca
    C = US
  Subject:
    Name:ssl-proxy.
    Serial Number:B0FFF0C2
    OID.1.2.840.113549.1.9.2 = ssl-proxy.
    OID.2.5.4.5 = B0FFF0C2
  CRL Distribution Point:
    http://sample.cisco.com/sub3ca.crl
  Validity Date:
    start date:18:04:09 UTC Jan 23 2003
    end date:21:05:17 UTC Dec 12 2003
    renew date:00:00:00 UTC Apr 1 2003
  Associated Trustpoints:tp-proxy1

CA Certificate
  Status:Available
  Certificate Serial Number:6D1E6B0F0000000000007
  Certificate Usage:Signature
  Issuer:
    CN = subtest
    C = US
  Subject:
    CN = sub3ca
    C = US
  CRL Distribution Point:
    http://sample.cisco.com/subtest.crl
  Validity Date:
    start date:22:22:52 UTC Mar 28 2003
    end date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:tp-proxy1

```

```

ssl-proxy# show crypto ca certificates 3tier-subcal
CA Certificate
  Status:Available
  Certificate Serial Number:29A47DEF0000000004E9
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
    C = US
  Subject:
    CN = subtest
    C = US
  CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:20:55:17 UTC Dec 12 2002
    end   date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:3tier-sub1

ssl-proxy# show crypto ca certificates 3tier-root
CA Certificate
  Status:Available
  Certificate Serial Number:7FD5B209B5C2448C47F77F140625D265
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
    C = US
  Subject:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
    C = US
  CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:00:05:32 UTC Jun 13 2002
    end   date:00:11:58 UTC Jun 13 2004
  Associated Trustpoints:3tier-root

```

## 証明書およびトラストポイントの確認

証明書とトラストポイントに関する情報を確認するには、EXEC モードで次の作業を行います。

|        | コマンド                                                                            | 目的                                                                  |
|--------|---------------------------------------------------------------------------------|---------------------------------------------------------------------|
| ステップ 1 | ssl-proxy(ca-trustpoint)# <b>show crypto ca certificates</b> [trustpoint_label] | 特定のトラストポイントに関連付けられた証明書または自分のすべての証明書、認証局の証明書、および登録局の証明書に関する情報を表示します。 |
| ステップ 2 | ssl-proxy(ca-trustpoint)# <b>show crypto ca trustpoints</b> [trustpoint_label]  | すべてのトラストポイントまたは特定のトラストポイントに関する情報を表示します。                             |

## 鍵および証明書の共有

SSL ドータカードは、複数の証明書による同一鍵の共有をサポートします。ただし、鍵のペアが 1 つ壊れると、すべての証明書が失効して取り替えられるので、望ましい方式ではありません。

プロキシサービスはさまざまなタイミングで追加されたり削除されたりするので、証明書の有効期限も異なります。認証局によっては、更新時に鍵のペアをリフレッシュすることが要求されます。証明書が 1 つの鍵のペアを共有している場合は、複数の証明書を同時に更新しなければなりません。通常、証明書ごとに専用の鍵のペアを与える方が証明書の管理は容易です。



複数のプロキシサービスと複数の SSL ドータカード間での証明書の共有について、SSL ドータカードからの制限はありません。複数のプロキシサービスに同じトラストポイントを割り当てることができます。

認証局はビジネス上の理由から、制限を課す場合があります（サーバファーム内で同じ証明書を使用できるサーバの数など）。証明書の共有に関して、契約または許諾が存在する場合があります。契約に関しては認証局または自社の法務担当者に問い合わせてください。

一部の Web ブラウザは、サーバの証明書に記された件名と URL に含まれているホスト名または IP アドレスを比較します。件名がホスト名または IP アドレスと一致しなかった場合、ダイアログボックスが表示され、証明書を確認してから受け入れることをユーザに要求します。このステップを回避するには、ホスト名または IP アドレスに基づいて証明書の共有を制限する必要があります。

## ルート認証局（信頼できるルート）の設定

信頼できるルートを設定するには、グローバル コンフィギュレーション モードから始めて次の作業を行います。

|        | コマンド                                                                   | 目的                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 1 | Router(config)# <b>crypto ca trusted-root</b> <i>name</i>              | 選択した名前でもルートを設定し、信頼できるルート コンフィギュレーション モードを開始します。                                                                                                                                                                                                                                                                                                                                                             |
| ステップ 2 | Router(ca-root)# <b>crl query url</b>                                  | (任意) LDAP <sup>1</sup> の URL で、設定したルートが公開した CRL について問い合わせます。                                                                                                                                                                                                                                                                                                                                                |
| ステップ 3 | Router(ca-root)# <b>exit</b>                                           | (任意) 信頼できるルート コンフィギュレーション モードを終了します。                                                                                                                                                                                                                                                                                                                                                                        |
| ステップ 4 | Router(config)# <b>crypto ca trustpoint</b> <i>name</i>                | (任意) 認証局アイデンティティ コンフィギュレーション モードを開始します。                                                                                                                                                                                                                                                                                                                                                                     |
| ステップ 5 | Router(ca-identity) # <b>crl optional</b>                              | (任意) スイッチが該当する CRL にアクセスできなかった場合でも、スイッチがほかのピアの証明書を受け入れるようにします。                                                                                                                                                                                                                                                                                                                                              |
| ステップ 6 | Router(ca-identity)# <b>exit</b>                                       | (任意) 認証局アイデンティティ コンフィギュレーション モードを終了します。                                                                                                                                                                                                                                                                                                                                                                     |
| ステップ 7 | Router(config)# <b>crypto ca trusted-root</b> <i>name</i>              | (任意) 信頼できるルート コンフィギュレーション モードを開始します。                                                                                                                                                                                                                                                                                                                                                                        |
| ステップ 8 | Router(ca-root)# <b>root {CEP url   TFTP server-hostname filename}</b> | <p>特定のアイデンティティと URL を指定して SCEP<sup>2</sup> を使用するか、または TFTP を使用して、ルートの証明書を取得します。</p> <div>  <p>(注) 認証局サーバが SCEP をサポートしない場合は、TFTP だけを使用します。</p> </div> <div>  <p>(注) TFTP を使用する場合は、ルート証明書のダウンロードが攻撃を受けないように、サーバを保護する必要があります。</p> </div> |
| ステップ 9 | Router(ca-root)# <b>root proxy url</b>                                 | ルートの証明書を受け取る HTTP プロキシサーバを定義します。                                                                                                                                                                                                                                                                                                                                                                            |

1. LDAP = Lightweight Directory Access Protocol

2. SCEP = Simple Certificate Enrollment Protocol

## コンフィギュレーションの保存



### 注意



RSA 鍵のペアが保存されるのは NVRAM だけです。 **copy system:running-config file\_system:** コマンドで他のファイルシステムを指定した場合、RSA 鍵はコンフィギュレーションには保存されません。



### ヒント

コンフィギュレーションを変更したときには、忘れずに変更した設定を保存してください。

NVRAM にコンフィギュレーションを保存する手順は、次のとおりです。

| コマンド                                                                           | 目的                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ssl-proxy# copy [/erase] system:running-config nvram:startup-config</pre> | <p>コンフィギュレーション、鍵のペア、および証明書を NVRAM に保存します。鍵のペアはプライベート コンフィギュレーション ファイルに保存され、各証明書は NVRAM にバイナリ ファイルとして保存されます。モジュールが起動時に、認証局に問い合わせで証明書を取得したり、自動登録したりする必要がなくなります。</p> <p> (注) セキュリティを確保するために、<b>/erase</b> キーワードを入力し、NVRAM の更新前にパブリックおよびプライベート コンフィギュレーション ファイルを消去する必要があります。<b>/erase</b> キーワードを入力しなかった場合、古いプライベート コンフィギュレーション ファイルに含まれていた鍵のペアが NVRAM に残ります。</p> <p> <b>注意</b> <b>/erase</b> キーワードを入力すると、NVRAM の現在のバッファとバックアップのバッファが消去されてから、実行コンフィギュレーションが NVRAM に保存されます。バッファの消去後、実行コンフィギュレーションを保存するまでの間に、電源障害または再起動が発生した場合は、両方のコンフィギュレーションが失われる可能性があります。</p> |



### (注)

NVRAM に多数のファイルが保存されている場合、この作業に最大で 2 分かかります。

NVRAM へのコンフィギュレーション自動バックアップ機能によって、最後に保存されたコンフィギュレーションのバックアップが作成されます。現在の書き込みプロセスが失敗すると、以前のコンフィギュレーションが自動的に復元されます。

## コンフィギュレーションが大きすぎる場合

プロキシサービス数が 256、証明書が 356 を超える大きすぎるコンフィギュレーションを保存すると、NVRAM の内容が壊れる可能性があります。

必ず、実行コンフィギュレーションにコピーしてから、NVRAM に保存してください。リモートサーバに実行コンフィギュレーション ファイルを保存すると、各証明書は 16 進ダンプとしてファイルに保存されます。実行コンフィギュレーション ファイルを実行コンフィギュレーションに戻してから NVRAM に保存すると、再び証明書が保存されますが、今度はバイナリ ファイルとして保存されます。ただし、リモートサーバからスタートアップ コンフィギュレーションに実行コンフィギュレーション ファイルを直接コピーした場合は、16 進ダンプとして保存された証明書も保存されるので、同じ証明書に対してコピーが 2 つできます。一方は 16 進ダンプ、他方はバイナリ ファイルです。この動作は不要であり、またリモート ファイルが非常に大きいと、NVRAM の内容が部分的に上書きされ、その結果、内容が壊れることがあります。

## 保存したコンフィギュレーションの確認

保存したコンフィギュレーションを表示する手順は、次のとおりです。

|        | コマンド                                  | 目的                               |
|--------|---------------------------------------|----------------------------------|
| ステップ 1 | ssl-proxy# <b>show startup-config</b> | スタートアップ コンフィギュレーションを表示します。       |
| ステップ 2 | ssl-proxy# <b>directory nvram:</b>    | NVRAM に保存されているファイルの名前とサイズを表示します。 |



(注) 最大数のプロキシ サービス (256) および証明書 (356) が設定されている場合、表示に最大で 7 分かかります。

## 保存したコンフィギュレーションの消去

保存したコンフィギュレーションを消去するには、次のどちらか一方の作業を行います。

| コマンド                                | 目的                                                           |
|-------------------------------------|--------------------------------------------------------------|
| ssl-proxy# <b>erase nvram:</b>      | スタートアップ コンフィギュレーションと鍵のペアを消去します。                              |
| ssl-proxy# <b>erase /all nvram:</b> | スタートアップ コンフィギュレーション、鍵のペア、証明書、およびその他のすべてのファイルを NVRAM から消去します。 |



(注) NVRAM に多数のファイルが保存されている場合、この作業に最大で 2 分かかります。



注意

保存したコンフィギュレーションを消去すると、自動的に作成された NVRAM 内のバックアップ コンフィギュレーションも消去されます。

## 鍵および証明書のバックアップ

鍵と証明書を NVRAM に保存するプロセスに割り込むイベント（電源障害など）が発生すると、保存作業中の鍵と証明書が失われる可能性があります。公開鍵および証明書は認証局から取得できますが、秘密鍵は回復できません。

セキュア サーバが利用できる場合は、PKCS12 ファイルに各トラストポイントをエクスポートすることによって、鍵のペアと対応する証明書のチェーンをバックアップします。さらに PKCS12 ファイルをインポートすると、鍵と証明書を復元できます。

## セキュリティに関する注意事項

鍵と証明書のバックアップ時には、次の注意事項に従ってください。

- PKCS12 ごとに、容易に推測できないパス フレーズを選択し、パス フレーズを安全確実に保存する必要があります。PKCS12 ファイルをクリア テキスト形式で保存してはなりません。
- バックアップ サーバを保護する必要があります。許可されたスタッフだけがバックアップ サーバにアクセスできるようにします。
- PKCS12 ファイルをインポートまたはエクスポートするときには、モジュールのコンソールに直接接続するか、または SSH セッションを使用します。パス フレーズはこのファイルに入力しなければなりません。
- ファイル転送には SCP を使用します。

## 鍵および証明書のモニタおよびメンテナンス

次の作業は任意です。

- [モジュールからの RSA 鍵の削除 \(p.8-35\)](#)
- [鍵および証明書の表示 \(p.8-36\)](#)
- [コンフィギュレーションからの証明書の削除 \(p.8-36\)](#)

## モジュールからの RSA 鍵の削除




### 注意

SSH 鍵を削除すると、モジュール上で SSH がディセーブルになります。SSH 鍵を削除する場合は、新しい鍵を生成してください。詳細については、「[SSH の設定 \(p.7-4\)](#)」を参照してください。

状況によっては、モジュールから RSA 鍵を削除する必要があります。たとえば、何らかの理由で RSA 鍵が壊れていることが明らかな場合で、なおかつ今後使用しない場合は、鍵を削除する必要があります。

モジュールからすべての RSA 鍵を削除するには、グローバル コンフィギュレーション モードで次の作業を行います。


| コマンド                                                               | 目的                                                                                                                                                                                                                                 |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssl-proxy(config)# crypto key zeroize rsa [key-label]</code> | すべての RSA 鍵のペアまたは特定の鍵のペアを削除します。                                                                                                                                                                                                     |
|                                                                    | <div style="display: flex; align-items: center;">  <div> <b>注意</b> 鍵を削除すると、その鍵に関連付けられているすべての証明書も削除されます。 </div> </div> |

モジュールから RSA 鍵を削除したあとで、2 つの手順が必要です。

- 認証局の管理者に、認証局側でモジュールの証明書を失効させるように要求します。証明書を最初に取得したときに、**crypto ca enroll** コマンドを使用してそのモジュール用に作成したチャレンジパスワードを提示する必要があります。
- 「[コンフィギュレーションからの証明書の削除](#)」(p.8-36) の説明に従って、コンフィギュレーションからトラストポイントを手動で削除します。

## 鍵および証明書の表示

鍵および証明書を表示するには、EXEC モードで次のどちらか一方の作業を行います。

| コマンド                                                                   | 目的                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssl-proxy# show crypto key mypubkey rsa</code>                   | モジュールの RSA 公開鍵を表示します。                                                                                                                                                                                                                  |
| <code>ssl-proxy# show crypto ca certificates [trustpoint_label]</code> | 証明書、認証局の証明書、および登録局の証明書に関する情報を表示します。                                                                                                                                                                                                    |
| <code>ssl-proxy# show running-config [brief]</code>                    | 公開鍵と証明書のチェーンを表示します。 <b>brief</b> キーワードを指定した場合、各証明書の 16 進ダンプは表示されません。                                                                                                                                                                   |
| <code>ssl-proxy# show ssl-proxy service proxy-name</code>              | 特定のプロキシ サービスに使用される証明書チェーンの鍵のペアおよびシリアル番号を表示します。                                                                                                                                                                                         |
|                                                                        | <div style="display: flex; align-items: center;">  <div> <b>(注)</b> <i>proxy-name</i> の値は、大文字と小文字が区別されます。 </div> </div> |

## コンフィギュレーションからの証明書の削除

モジュールでは、独自の証明書と認証局の証明書を保存します。モジュール上に保存されている証明書を削除できます。

モジュール コンフィギュレーションから証明書を削除するには、グローバル コンフィギュレーション モードで次の作業を行います。

| コマンド                                                                     | 目的         |
|--------------------------------------------------------------------------|------------|
| <code>ssl-proxy(config)# no crypto ca trustpoint trustpoint-label</code> | 証明書を削除します。 |



## プロキシ サービスへの証明書の割り当て

**certificate rsa general-purpose trustpoint** *trustpoint\_label* サブコマンド (**ssl-proxy service proxy\_service** コマンド下) を入力すると、特定のプロキシ サービスに証明書を割り当てることができます。

**certificate rsa general-purpose trustpoint** サブコマンドは、プロキシ サービスに対して繰り返し入力できます。

トラストポイント ラベルが変更されると、移行時にプロキシ サービスがサービスから除外されます。既存の接続では、その接続が終了するか切断されるまで、古い証明書が引き続き使用されます。新しい接続では新しいトラストポイントからの証明書が使用され、サービスが再び利用可能になります。

ただし、新しいトラストポイントに証明書がまだない場合は、サービスの動作ステータスがダウンのままです。新しい証明書が利用できるようになるまで、新しい接続は確立されません。

**no certificate rsa general-purpose trustpoint** サブコマンドを入力して証明書を削除した場合、既存の接続では、その接続が終了するか切断されるまで、削除した証明書が引き続き使用されます。証明書は廃棄されても、すべての接続が終了するか切断されるまでは、プロキシ サービスから削除されません。

次に、プロキシ サービスにトラストポイントを割り当てる例を示します。

```
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# virtual ip 10.1.1.2 p tcp p 443
ssl-proxy(config-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-1
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-1
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:tp-1
    Serial Number:3C2CD2330001000000DB
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#
```

次に、プロキシ サービス用のトラストポイントを変更する例を示します。



(注)

既存の接続では、その接続が終了するまで、古い証明書が引き続き使用されます。サービスの動作ステータスはアップからダウンに変化し、再びアップになります。新しい接続では新しい証明書が使用されます。

```
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:70FCBFEC000100000D65
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Obsolete certificate chain in use for old connections:
  Server Certificate:
    Key Label:tp-1
    Serial Number:3C2CD2330001000000DB
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#
```

## 証明書の更新

認証局によって、新しい鍵のペアを生成して証明書を更新するように要求することもあれば、有効期限が切れた証明書の鍵のペアを使用して証明書を更新できることもあります。CSM-S では、どちらの場合もサポートされます。

SSL サーバの証明書は通常、1 年または 2 年で有効期限が切れます。証明書の期限を上手に延長することによって、突発的なサービス停止を回避できます。

次の例では、プロキシ サービス s2 にトラストポイント t2 が割り当てられています。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint t2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#

ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
```

次の例では、トラストポイント t2 の鍵のペアが更新され、Cisco IOS データベースから古い証明書が削除されます。プロキシ サービス s2 に対して、スムーズな移行が自動的に開始されます。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto key generate rsa general-key k2 exportable
% You already have RSA keys defined named k2.
% Do you really want to replace them? [yes/no]:yes
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
ssl-proxy(config)#end
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in graceful rollover, being renewed:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Server certificate in graceful rollover
Admin Status:up
Operation Status:up
```

次の例では、トラストポイント t2 が再登録されるまで、既存の接続と新しい接続で古い証明書が使用されます。トラストポイント t2 の再登録後は、新しい接続で新しい証明書が使用され、既存の接続ではその接続が終了するまで、古い証明書が引き続き使用されます。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto ca enroll t2
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=host1.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:00000000
% The IP address in the certificate is 10.1.1.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

Fingerprint: 6518C579 A0498063 C5795057 A6170 075

ssl-proxy(config)# end
*Sep 24 15:19:34.339:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority

ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:2475A2FC000100000D4D
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Obsolete certificate chain in use for old connections:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
```

次の例では、廃棄された証明書が既存のすべての接続が終了したあとで削除されます。

```
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:2475A2FC000100000D4D
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
```

## 証明書の自動更新および登録の設定

自動登録を設定すると、コンフィギュレーションでパラメータを使用している認証局に、モジュールが証明書を自動的に要求します。

有効期間の一定の割合が経過すると、証明書が自動的に更新されるように設定できます。たとえば、有効期間が 300 日の証明書の場合、*renewal\_percent* を 80 として指定すると、証明書の有効期間が開始されてから 240 日後に、証明書が自動的に更新されます。



(注)

認証局の証明書は、自動登録または自動更新する前にデータベースに保存する必要があります。また、自動登録を設定する前にトラストポイントの認証が必要です。さらに、トラストポイント用に SCEP 登録の URL を設定します。

自動登録と自動更新をイネーブルにして、タイマー情報を表示する手順は、次のとおりです。

|        | コマンド                                                                                          | 目的                                                                                                                                                                                                                  |
|--------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# <b>crypto ca trustpoint</b> trustpoint-label</code>                  | トラストポイントを宣言します。                                                                                                                                                                                                     |
| ステップ 2 | <code>ssl-proxy(ca-trustpoint)# <b>auto-enroll</b> {renewal_percent <b>regenerate</b>}</code> | <p>特定のトラストポイントに対して、自動更新と自動登録をイネーブルにします。</p> <div> <p>(注) <i>renewal_percent</i> の有効な値は 0（1 分以内に登録）～ 100 です。</p> </div> <div> <p>(注) <b>regenerate</b> キーワードを指定すると、指定した鍵がすでに存在する場合でも、証明書に対応する新しい鍵が生成されます。</p> </div> |
| ステップ 3 | <code>ssl-proxy# <b>show crypto ca timers</b></code>                                          | 各タイマーが満了するまでの残り時間を表示します。                                                                                                                                                                                            |

次に、自動登録と自動更新をイネーブルにする例を示します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto ca trustpoint tk21
ssl-proxy(ca-trustpoint)# auto-enroll 90
ssl-proxy(ca-trustpoint)# end
ssl-proxy# show crypto ca timers
PKI Timers
|          44.306
|          44.306  RENEW  tp-new
|255d 5:28:32.348  RENEW  tk21
ssl-proxy#
```

## 鍵および認証に関する履歴のイネーブル化

**ssl-proxy pki history** コマンドを入力すると、SSL プロキシ サービスの鍵と証明書に関する履歴がイネーブルになります。この履歴では、プロキシ サービスに対応する鍵のペアと証明書が追加または削除されるたびに、レコードが作成されます。

**show ssl-proxy certificate-history** コマンドを入力すると、レコードが表示されます。各レコードにサービス名、鍵のペア名、生成またはインポート時刻、トラストポイント名、証明書の件名、証明書の発行元名、シリアル番号、および日付が記録されます。

最大 512 のレコードをメモリに保存できます。レコードごとに Syslog メッセージが生成されます。512 のレコード限度に達すると、最古のレコードが削除されます。

鍵と証明書の履歴をイネーブルにして、レコードを表示する手順は、次のとおりです。

|        | コマンド                                                                               | 目的                                         |
|--------|------------------------------------------------------------------------------------|--------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# ssl-proxy pki history</code>                              | 鍵と証明書の履歴をイネーブルにします。                        |
| ステップ 2 | <code>ssl-proxy# show ssl-proxy certificate-history [service proxy_service]</code> | すべてのサービスまたは特定のサービスについて、鍵と証明書の履歴レコードを表示します。 |

次に、鍵と証明書の履歴をイネーブルにして、特定のプロキシ サービスに対応するレコードを表示する例を示します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)#ssl-proxy pki history
ssl-proxy(config)#end
ssl-proxy# show ssl-proxy certificate-history service s2
Record 1, Timestamp:00:00:22, 17:44:18 UTC Sep 29 2002
  Installed Server Certificate, Index 0
  Proxy Service:s2, Trust Point:t2
  Key Pair Name:k2, Key Usage:RSA General Purpose, Not Exportable
  Time of Key Generation:06:29:08 UTC Sep 28 2002
  Subject Name:CN = host1.cisco.com, OID.1.2.840.113549.1.9.2 = ssl-proxy.cisco.com,
OID.1.2.840.113549.1.9.8 = 10.1.1.1
  Issuer Name:CN = TestCA, OU = Lab, O = Cisco Systems, L = San Jose, ST = CA, C =
US,
EA =<16> simpson-pki@cisco.com
  Serial Number:3728ADCD000100000D4F
  Validity Start Time:15:56:55 UTC Sep 28 2002
  End Time:16:06:55 UTC Sep 28 2003
  Renew Time:00:00:00 UTC Jan 1 1970
  End of Certificate Record
Total number of certificate history records displayed = 1
```

## ピア証明書のキャッシング

SSL ドータカードを設定することによって、認証済みサーバおよびクライアント（ピア）の証明書をキャッシュできます。認証済みピアの証明書をキャッシュすると、SSL ドータカードは同じ証明書をもう一度認証しなくてすむので、時間を節約できます。SSL ドータカードは、指定されたタイムアウト インターバルの間に同じピア証明書を受信し、確認オプション（署名のみ、または全部）が一致した場合に、キャッシュ内の証明書情報を使用します。



(注)

**verifying all** コマンドを入力した場合、CRL 検索または ACL フィルタリングの結果が指定のタイムアウト インターバルの間に変わることがあるので、SSL ドータカードは拒否すべき証明書を誤って受け入れてしまう可能性があります。たとえば、SSL ドータカードがピア証明書をキャッシュに保存し、さらに指定のタイムアウトまでに、その証明書が記載された更新後の CRL をダウンロードするといったことが考えられます。

SSL ドータカードが多数の証明書をたびたび受信し、このリスクが許容される場合は、認証済みのピア証明書をキャッシュすることによって、オーバーヘッドを軽減できます。

たとえば、サイト間の VPN 環境において、2 つの SSL ドータカードがフルハンドシェイク時に、相互に相手の証明書を認証する場合は、キャッシングが適用可能です。署名のみの確認と認証済みピア証明書のキャッシングを組み合わせることによって、最良のパフォーマンスが得られます。

指定のタイムアウト インターバルの間に期限切れになるピア証明書はキャッシュされません。SSL ドータカードは、署名のみのオプションとすべて確認のオプションに別々のキャッシュ エントリを使用します。確認オプションの一致は、キャッシュのヒット条件の 1 つです。

さまざまなプロキシ サービスでさまざまな時期に同じピア証明書を受信する可能性があり、さらにプロキシ サービスごとに専用の認証局プールがあるので、ピア証明書の発行元が以前のプロキシ サービスの認証局プールに属し、現在のプロキシ サービスの認証局プールには含まれていない状況もありえます。この状況はキャッシュ ミスとみなされ、ピア証明書が検証されます。

認証済みのピア証明書をキャッシュする手順は、次のとおりです。

| コマンド                                                                                            | 目的                                                                                                                                                                                    |
|-------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssl-proxy(config)# <b>ssl-proxy pki cache</b><br/><b>size size timeout minutes</b></code> | 証明書キャッシュ パラメータを設定します。<br><br><i>size</i> のデフォルト値は 0（ディセーブル）です。有効な値は 0 ～ 5000 エントリです。 <i>minutes</i> のデフォルト値は 15 分です。有効な値は 1 ～ 600 分です。<br><br>キャッシュを消去するには、キャッシュ サイズまたはタイムアウト値を変更します。 |

## 証明書の有効期限に関する警告の設定

証明書の有効期限が切れた時点で、または期限切れまでの残り時間が指定された値になった時点で、警告メッセージが記録されるように SSL ドータカードを設定できます。証明書の有効期限に関する警告をイネーブルにすると、SSL ドータカードは 30 分おきに下記について期限情報をチェックします。

- あらゆるプロキシ サービス
- プロキシ サービスに関連付けられている認証局の証明書
- 信頼できる認証局のプールに割り当てられているすべての認証局トラストポイント

指定できるタイム インターバルは 1 ～ 720 時間です。




(注)

SSL ドータカードは、どの証明書が記録されたかについて、情報を保存します。0 を指定すると、警告がディセーブルになり、それまで記録されていた警告メッセージの内部メモリが消去されます。1 ～ 720 時間の範囲でタイム インターバルを指定すると、ロギングプロセスが再開されます。ロギングの再開直後は、ログ メッセージが表示されないことがあります。最初のログ メッセージを確認できるまでに最長で 30 分待つ場合もあります。

指定したインターバルの間に証明書の有効期限が切れる場合、またはすでに期限が切れている場合、SSL ドータカードがその証明書について記録する警告メッセージは 1 つだけです。

さらに、CISCO-SSL-PROXY-MIB という証明書の有効期限トラップをイネーブルにすると、プロキシ サービスの証明書の有効期限に関する警告が記録されるたびに、トラップが発行されます。SNMP トラップの設定の詳細については、「[SNMP トラップの設定](#)」(p.7-26) を参照してください。

証明書の有効期限に関する警告をイネーブルにして、警告のインターバルを設定する手順は、次のとおりです。

| コマンド                                                                                            | 目的                                                                                                                                                                      |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ssl-proxy(config)# <b>ssl-proxy pki certificate check-expiring interval interval</b></pre> | <p>証明書の有効期限に関する警告をイネーブルにして、警告のインターバルを設定します。<i>interval</i> のデフォルト値は 0 (ディセーブル) です。有効な値は 1 ～ 720 時間です。</p>                                                               |
|                                                                                                 | <p> (注) 0 を指定すると、警告がディセーブルになり、それまで記録されていた警告メッセージの内部メモリが消去されます。SNMP トラップは送信されません。</p> |

次に、証明書の有効期限に関する警告をイネーブルにして、警告のインターバルを設定する例を示します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 36
*Nov 27 03:44:05.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate
expiration
warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end
```



次に、それまで記録されていた警告メッセージの内部メモリを消去し、ロギングプロセスを再開する例を示します。

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 0
*Nov 27 03:44:15.207:%STE-6-PKI_CERT_EXP_WARN_DISABLED:Checking of certificate
expiration has been disabled.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 1
*Nov 27 03:44:16.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate
expiration
warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end
```

次に、有効期限が切れるか、すでに切れたプロキシ サービス証明書に対応する、Syslog メッセージの例を示します。

```
Jan 1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CERT_EXPIRING:A proxy service certificate
is
going to expire or has expired at this time:20:16:26 UTC Sep 5 2004, proxy service:s7,
trustpoint:tk21.
```

次に、有効期限が切れるか、すでに切れた 1 つまたは複数のプロキシ サービスに対応する、認証局の証明書に関する Syslog メッセージの例を示します。

```
Jan 1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CA_CERT_EXPIRING:A CA certificate is
going to
expire or has expired at this time:22:19:38 UTC Mar 4 2004, subject name:CN =
ExampleCA,
OU = Example Lab, O = Cisco Systems, L = San Jose, ST = CA, C = US, EA =
example@cisco.com, serial number:313AD6510D25ABAE4626E96305511AC4.
```

次に、有効期限が切れるか、すでに切れた、信頼できる認証局プールに割り当てられている認証局証明書に関する Syslog メッセージの例を示します。

```
Jan 1 00:00:18.971:%STE-4-PKI_CA_POOL_CERT_EXPIRING:A CA certificate in a CApool is
going
to expire or has expired at this time:22:19:38 UTC Mar 4 2004, CA pool:pool2,
trustpoint:tp1-root.
```

## 認証局の設定

ここでは、クライアント / サーバ認証の設定方法について説明します。

- クライアントの証明書の認証 (p.8-47)
- サーバの証明書の認証 (p.8-50)
- CRL (p.8-54)
- Certificate Security Attribute-Based Access Control (p.8-59)

クライアントまたはサーバの証明書の認証を設定する場合は、**signature-only** または **all** として検証形式を指定する必要があります。どちらのオプションでも、認証する各証明書の有効性の開始時刻と終了時刻がチェックされます。開始時刻にまだ達していない場合、または終了時刻をすでに過ぎている場合、SSL ドータカードはその証明書を受け付けません。

**verify signature-only** コマンドを入力すると、SSL ドータカードはピア証明書から次の証明書（前の証明書の発行元）までの証明書チェーンを検証し、さらに次の証明書を検証し、次に示す条件の 1 つが満たされるまでこれを繰り返します。

- 証明書が信頼できる認証局によって発行されているか、または証明書そのものが信頼できる認証局の証明書と一致し、なおかつ信頼できる認証局がプロキシ サービスに割り当てられた認証局プールに含まれています。この場合、チェーンは受け入れられます。チェーンの残りの部分を検証する必要はありません。
- チェーンの末尾に到達し、チェーンの最終証明書が信頼できる認証局の発行したものではありません。この場合、チェーンは拒否されます。

**verify all** コマンドを入力すると、SSL ドータカードは証明書チェーンを順番に分類し、無関係な証明書や重複する証明書を無視します。SSL ドータカードは、分類されたチェーンで最上位に位置する証明書が、信頼できる認証局の発行したものであるか、または信頼できる認証局の証明書と一致するかどうかを判別します。

SSL ドータカードが最上位の証明書を信頼できないとみなした場合、そのチェーンは拒否されます。

SSL ドータカードが最上位の証明書を信頼できるとみなした場合、SSL ドータカードはチェーン内の証明書ごとに次の動作を実行します。

- 各証明書の署名を検証します。
- 証明書が 1 つまたは複数のトラストポイントに関連付けられている場合、SSL ドータカードはそこから 1 つのトラストポイントを選択します。そのトラストポイントに対応する CRL と ACL マップの設定に基づいて、SSL ドータカードは失効と証明書の属性フィルタリングを実行します。CRL または ACL のチェックで証明書が拒否された場合、SSL ドータカードはそのチェーンを拒否します。
- 証明書が X509 バージョン 3 の認証局の証明書だった場合、SSL ドータカードは Basic Constraints 拡張フィールドが存在し、なおかつ有効かどうかを確認します。Basic Constraints 拡張フィールドが存在しない場合、または有効ではなかった場合、そのチェーンは拒否されます。

署名だけを検証する場合、検証プロセスではチェーン内で最小限の数の証明書について、有効性と署名だけをチェックします。すべてを検証する場合は、さらに多くのチェックが行われ、受け取ったすべての証明書について有効性を検査しますが、それだけ時間がかかり、消費する CPU 時間も多くなります。

CLI コマンドを入力して CRL をダウンロード、更新すると、リアルタイム遅延を短縮できます。ただし、CRL 検索は時間のかかるプロセスです。CRL については、「[CRL](#)」(p.8-54)を参照してください。

## クライアントの証明書の認証

SSL ドータカードを SSL サーバとして設定した場合、SSL クライアントを認証するように SSL ドータカードを設定できます。この場合、SSL ドータカードは SSL クライアントに証明書の要求して認証します。

SSL クライアントを認証する場合、SSL ドータカードは下記を検証します。

- あるレベルの証明書が次のレベルの発行元によって正しく署名されている。
- 証明書チェーン内の少なくとも 1 つの発行元の証明書が SSL プロキシ サービスによって信頼されている。
- 証明書チェーンの中に、CRL に含まれているもの、または ACL で拒否されたものが 1 つもない。

SSL クライアントの証明書を検証するには、信頼できる認証局のリスト（認証局プール）を指定して、SSL ドータカードを設定します。信頼できる認証局プールは、データベースに登録されている信頼できる認証局のサブセットです。SSL ドータカードが信頼するのは、認証局プールに設定されている認証局が発行した証明書だけです。



(注) プロキシ サービスを動作可能にするには、認証局プールに証明書のあるトラストポイントが 1 つまたは複数なければなりません。認証局プールのどのトラストポイントにも証明書がないと、プロキシ サービスは自動的にダウンになります。






(注) 階層構造の特定のレベルの証明書が認証局プールに含まれていないと、認証に失敗する可能性があります。認証で署名のみの検証を行う場合に、このタイプのエラーを回避して効率を改善するには、すべてのレベルの従属認証局をルート認証局とともに認証局プールに追加します。



(注) 認証局のトラストポイントが削除された場合は、信頼できる認証局プールから対応するトラストポイントを削除する必要があります。

クライアント認証を設定する手順は、次のとおりです。

|        | コマンド                                                                                        | 目的                                                                                                                                                |
|--------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# ssl-proxy pool<br/>ca_pool_name</code>                             | 認証局プールを作成します。<br><br>(注) 最大 8 つのプールを作成できます。                    |
| ステップ 2 | <code>ssl-proxy(config-ssl-proxy)# ca<br/>trustpoint ca_trustpoint_label<sup>1</sup></code> | プールに信頼できる認証局を追加します。<br><br>(注) 1 つのプールに最大 16 の信頼できる認証局を追加できます。 |
| ステップ 3 | <code>ssl-proxy(config-ssl-proxy)# exit</code>                                              | コンフィギュレーション モードに戻ります。                                                                                                                             |
| ステップ 4 | <code>ssl-proxy(config)# ssl-proxy service<br/>proxy_name</code>                            | SSL サーバのプロキシ サービス名を定義します。                                                                                                                         |

|        | コマンド                                                                                                                 | 目的                                                                                                                                                                                                                            |
|--------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 5 | <code>ssl-proxy(config-ssl-proxy)#<br/>trusted-ca ca_pool_name</code>                                                | 信頼できる認証局プールとプロキシ サービスを関連付けます。                                                                                                                                                                                                 |
| ステップ 6 | <code>ssl-proxy(config-ssl-proxy)#<br/>authenticate verify<br/>{signature-only<sup>2</sup>   all<sup>3</sup>}</code> | クライアントの証明書認証をイネーブルにして、検証の形式を指定します。<br><br> (注) <b>signature-only</b> キーワードを指定すると、署名だけが検証されます。CRL の検索または ACL の照合は行われません。デフォルトは <b>all</b> です。 |
| ステップ 7 | <code>ssl-proxy(config-ssl-proxy)# exit</code>                                                                       | コンフィギュレーション モードを終了します。                                                                                                                                                                                                        |

1. SSL ドータカードは、最大 8 レベルの認証局をサポートします。認証局プールにすべてのレベルの認証局を追加するか、または少なくともルート認証局を追加することを推奨します。
2. 署名のみを検証する場合、信頼できる認証局プール内の信頼できる認証局トラストポイントの 1 つに対応するレベルで認証が中止されます。
3. すべての検証する場合は、証明書チェーンの最上位の発行元が信頼できる認証局トラストポイントとして設定されていなければなりません。SSL ドータカードは、ピア証明書チェーンに含まれるすべての証明書を認証し、最上位の認証局でのみ停止します。最上位の認証局に対応する認証局トラストポイントがなければなりません。また、このトラストポイントを認証する必要があります。

次に、署名だけを検証する、クライアントの証明書認証を設定する例を示します。

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate rootca
```

```
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
```

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcmlpYTERMA8GA1UEBxMic2Fu
IGpvc2UxZDjAMBgNVBAoTBWNPc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEeMTIxNDgwM1oXDTEzMTEeMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbg1mb3JuaWEeETAPBgNV
BACgTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYD
VQDEdExdzaW1wc29uLWRLdnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVgQNU0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWCwYDVDR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVRO0BBYEFYGLUBTKNd9EgUonHnoSvBHg0axMIGX
BgNVHR8EgY8wYywwQ6BBOD+GPWh0dHA6Ly9jaXNjb3JuaWEeETAPBgNV
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcm9vdC1DQ55jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xZDZlJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBMBNlNyBDAQBgkrBgEeAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBQe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
```

```
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

```
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-auth-sig-only
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

次に、すべてを検証する、クライアントの証明書認証を設定する例を示します。

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate rootca
```

Enter the base 64 encoded CA certificate.  
End with a blank line or the word "quit" on a line by itself

```

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFAD
MQswCQYDVQQGEwJVVuZETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBmMiczFu
IGpvc2UxXjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBxb24tZGV2dGZvZDc1yb290LUNBMB4XDTAzMTEgMTIxNDgwMl0XDTEzMTEgMTIx
NzcyOjEwMDVmdGZvZDc1UEBhMCCVMBxwEzARBGNVBAgTMMNhbG1mb3JuaUEEaETAPBg
BACtCHNhb3N1MQ4wDAYDVQQKEwVjXjNjZEMMAoGA1UEC3MDAHNzMSAwHgYD
VQDExdzawlwc29uLWRLdnRlc3Qtcm9vdC1DQTBcMA0GCsGqGS1b3DQEBAQUAA0SA
MEgCQQCWEibAnUlVgQNU0Wb94qnH8iFKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm/raKJ/7ZAGMBAAGjgewwgekWcYDVR0PBAQDAGHGMASG
A1UdEWEb/wQFMAMBAf8whQYDVR0BBYEFYGLUBTlKND9EgUonHnoSvbnHg0axMIGX
BgNVHR8EgY8wYywwGzBBDO+GPwh0DHAA6Ly9jaXNjb3N1b3R0b2h2bWbIvQ2VydE
Vucm9sb3C9Zawlwc29uLWRLdnRlc3Qtcm9vdC1DQS5jcmwwRaBDOEGGP2ZpbGU6
Ly9cXGNpc2NvLWlw4ajZvaHBuc1dXZXJRW5yb2xsXHNpbXBxb24tZGV2dGZvZDc1yb290
LUNBMLNyb2DAQBgkrBgEeAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
Yjale1GZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pC6eIO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

```

```
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29B891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

```
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-auth-verify-all
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

## サーバの証明書の認証

SSL ドータカードを SSL クライアント（バックエンド暗号化用など）として設定すると、SSL ドータカードはつねに SSL サーバを認証することになります。

SSL サーバを認証する場合、SSL ドータカードは下記を検証します。

- あるレベルの証明書が次のレベルの発行元によって正しく署名されている。
- 証明書チェーン内の少なくとも 1 つの発行元の証明書が SSL プロキシ サービスによって信頼されている。
- 証明書チェーンの中に、CRL に含まれているもの、または ACL で拒否されたものが 1 つもない。


SSL ドータカードはデフォルトで、証明書があり、CRL に含まれていなくて、なおかつ ACL によって拒否されていない、あらゆる認証局のトラストポイントが発行したあらゆる証明書を受け付けます。

任意で、信頼できる認証局プールを作成し、プロキシ サービスに関連付けることができます。この場合、SSL ドータカードはそのプールに含まれている認証局が発行した証明書だけを受け付けます。

**authenticate verify signature-only** コマンドを入力することによって、署名だけの検証を選択することもできます。署名を検証する場合、CRL チェックと ACL チェックは省略されます。**signature-only** オプションを指定するには、信頼できる認証局プールを設定する必要があります。

サーバの証明書認証を設定する手順は、次のとおりです。

|        | コマンド                                                                                        | 目的                                                                                                                                                                                     |
|--------|---------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# ssl-proxy pool<br/>ca_pool_name</code>                             | 認証局プールを作成します。<br><br>(注) 最大 8 つのプールを作成できます。                                                         |
| ステップ 2 | <code>ssl-proxy(config-ssl-proxy)# ca<br/>trustpoint ca_trustpoint_label<sup>1</sup></code> | プールに信頼できる認証局を追加します。<br><br>(注) 1 つのプールに最大 16 の信頼できる認証局を追加できます。                                      |
| ステップ 3 | <code>ssl-proxy(config-ssl-proxy)# exit</code>                                              | コンフィギュレーション モードに戻ります。                                                                                                                                                                  |
| ステップ 4 | <code>ssl-proxy(config)# ssl-proxy service<br/>proxy_name client</code>                     | SSL クライアントのプロキシ サービス名を定義します。<br><br>(注) <b>client</b> キーワードを入力することによって、SSL クライアント プロキシ サービスを設定します。  |
| ステップ 5 | <code>ssl-proxy(config-ssl-proxy)#<br/>trusted-ca ca_pool_name</code>                       | (任意) 認証局プールとプロキシ サービスを関連付けます。<br><br>(注) ステップ 6 で <b>signature-only</b> を指定する場合、認証局プールを設定する必要があります。 |

|        | コマンド                                                                                                               | 目的                                                                                                                                                                                                                                                                                       |
|--------|--------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ 6 | <pre>ssl-proxy(config-ssl-proxy)#<br/>authenticate verify<br/>{signature-only<sup>2</sup>   all<sup>3</sup>}</pre> | <p>(任意) サーバの証明書認証をイネーブルにして、検証の形式を指定します。</p> <p> (注) <b>signature-only</b> キーワードを指定すると、署名だけが検証されます。CRL の検索または ACL の照合は行われません。<b>signature-only</b> を指定する場合は、認証局プールを設定する必要があります。デフォルトは <b>all</b> です。</p> |
| ステップ 7 | <pre>ssl-proxy(config-ssl-proxy)# exit</pre>                                                                       | コンフィギュレーション モードを終了します。                                                                                                                                                                                                                                                                   |

1. SSL ドータカードは、最大 8 レベルの認証局をサポートします。認証局プールにすべてのレベルの認証局を追加するか、または少なくともルート認証局を追加することを推奨します。
2. 署名のみを検証する場合、信頼できる認証局プール内の信頼できる認証局トラストポイントの 1 つに対応するレベルで認証が中止されます。
3. すべてのを検証する場合は、証明書チェーンの最上位の発行元が信頼できる認証局トラストポイントとして設定されていなければなりません。SSL ドータカードは、ピア証明書チェーンに含まれるすべての証明書を認証し、最上位の認証局でのみ停止します。最上位の認証局に対応する認証局トラストポイントがなければなりません。また、このトラストポイントを認証する必要があります。

次に、署名だけを検証する、サーバの証明書認証を設定する例を示します。

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate rootca

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcmlpYTERMA8GA1UEBxMlcn2Fu
IGpvc2UxZDjAMBgNVBAoTBWVnc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTIwNDgwMloXDTEzMTEwMTIx
NTczOVowdTELMAKGA1UEBhMCMVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEwETAPBgNV
BACTCmNhbG1mb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECjMDaHNSMSAwHgYD
VQDEExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQUNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgewkCwYDVVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVROBBYEFcyGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBOD+GPWh0dHA6Ly9jaXNjb3NlOGo2b2hwb3NlV2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQTS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBMLNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqelwy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sig-only client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```





次に、すべてを検証するサーバの証明書認証を設定する例を示します。SSL ドータカードは、クライアントとして設定され、要求された場合に SSL サーバに証明書を送信します。

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate rootca
```

Enter the base 64 encoded CA certificate.  
End with a blank line or the word "quit" on a line by itself

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcmlpYTERMA8GA1UEBxMlcmFu
IGpvc2UxZDjAMBGBA0TBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTIwNDgwMl0XDTEzMTEwMTIw
NTczOVowdTELMAKGA1UEBhMCVVMxEzARBgNVBAGTCmNhbg1mb3JuaWEwETAPBgNV
BACTCmNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECmMDaHhMSAwHgYD
VQDEExdzaW1wc29uLWRLdnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQUN0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgewkCwYDVR0PBAQDAGHMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVROBBYEFYCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBOD+GPWh0dHA6Ly9jaXNjbys0G02b2hwbNivQ2VydEVu
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcm9vdC1DQSBjcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBMB4wDQYJKoZIhvcNAQEBGQ8gAgEAAQIBADANBgkqhkiG9w0BAQUFAANBACBqelwy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
```

```
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

```
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sending-client-cert client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.5 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.3 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

## CRL

Certificate Revocation List (CRL; 証明書失効リスト) は、信用できなくなった証明書を特定したタイムスタンプ付きのリストです。失効した各証明書は、証明書のシリアル番号で CRL に記載されます。関係するピア装置が証明書を使用する場合、その装置は証明書の署名と有効性を確認するだけでなく、証明書のシリアル番号が CRL に登録されていないかどうか調べる。



(注) CRL をダウンロードして使用すると、時間がかかり、CPU の消費も大幅に増えます。

ここでは、CRL のダウンロード方法および設定方法について説明します。

- [CRL のダウンロード \(p.8-55\)](#)
- [CRL オプションの設定 \(p.8-56\)](#)
- [CRL の更新 \(p.8-57\)](#)
- [X.500 CDP 情報の入力 \(p.8-57\)](#)
- [CRL の手動入力 \(p.8-58\)](#)
- [CRL 情報の表示 \(p.8-59\)](#)
- [CRL の削除 \(p.8-59\)](#)

## CRL のダウンロード

検証する証明書に CRL Distribution Point (CDP) 拡張フィールドが含まれている場合、モジュールはダウンロードパスとしてその CDP を使用します。SSL ドータカードは、3 種類の CDP をサポートします。

- HTTP URL  
例 : `http://hostname/file.crl`
- X.500 Distinguished Name (DN)  
例 : `CN=CRL,O=cisco,C=us`
- Lightweight Directory Access Protocol (LDAP) URL  
例 : `ldap://hostname/CN=CRL,O=cisco,C=us`

証明書に CDP がない場合、SSL ドータカードは証明書と関連付けられているトラストポイントを探します。SSL ドータカードが対応するトラストポイントを 1 つまたは複数見つけた場合、SSL ドータカードはそれらのトラストポイントのうちの 1 つを使用し、さらに SCEP 登録 URL を使用してダウンロードパスとプロトコルを決定します。SCEP 登録 URL がない場合は、検証エラーになります。



(注)

CRL 要求に SCEP を使用する場合は、鍵のペアとトラストポイントを関連付ける必要があります。この目的では既存の鍵のペアをどれでも割り当てることができます。鍵のペアは、CRL ダウンロード要求の署名に使用します。

SSL ドータカードは次の理由で、ルート認証局の証明書については CRL 検索を行いません。

- ルート認証局の証明書の多くは、CDP 拡張フィールドが含まれていません。認証局が CRL 要求に関して SCEP をサポートしない場合、CRL のダウンロードは失敗します。
- CRL にはルート認証局が署名します。ルート認証局が取り消された場合は、CRL が通常は無効になります。失効したルート認証局に関連付けられたすべてのトラストポイントは、失効が判明した時点で速やかにデータベースから削除する必要があります。

ダウンロードパスが不明の場合、またはダウンロード動作が失敗した場合、ピアの証明書チェーンが拒否されます。

モジュールは CRL をダウンロードしたあとで、証明書のシリアル番号が CRL に含まれているかどうかを調べます。検証している証明書のシリアル番号が CRL に含まれていた場合、ピアの証明書チェーンは拒否されます。



- (注) 認証局の証明書のいくつかが信頼できる場合でも、ピアの証明書チェーンに含まれている 1 つまたは複数の証明書が原因で、CRL 検索に失敗することがあります（たとえば、認証局の証明書がインポート後に取り消された場合、またはその認証局の証明書用にダウンロードした CRL の有効期限が切れ、更新された CRL をダウンロードできなかった場合）。

## CRL オプションの設定



- (注) 複数のトラストポイントをルートまたは従属認証局の証明書に関連付けることができます。証明書の認証時に、この中から任意のトラストポイントを選択して CRL の設定を決定できます。一貫性のある認証結果を得るには、これらのトラストポイントのすべてに同じ CRL 設定を与える必要があります。

SSL ドータカードはデフォルトで、証明書を検証するためにトラストポイントが選択されている場合、つねに CRL 検索を実行します。CRL がデータベースにない場合、または有効期限が切れた場合、SSL ドータカードは CRL をダウンロードして、あとから使用できるようにデータベースに保存します。CRL のダウンロードに失敗した場合、SSL ドータカードは検証中の証明書を拒否します。

CRL 検索には、次の 2 つのオプションを設定できます。

- best-effort

SSL ドータカードがデータベースで CRL を見つけ、その CRL の有効期限が切れていなかった場合、SSL ドータカードは CRL 検索を実行します。SSL ドータカードが CRL を見つけられなかった場合、SSL ドータカードは CRL のダウンロードを試みます。ただし、CRL のダウンロードに失敗した場合、SSL ドータカードは証明書を受け付けます。

- optional

SSL ドータカードがデータベースで有効期限の切れていない CRL を見つけた場合、SSL ドータカードは CRL 検索を実行します。SSL ドータカードが CRL を見つけられなかった場合、SSL ドータカードは証明書を受け付けます。SSL ドータカードは CRL のダウンロードを試みません。

CRL オプションの設定手順は、次のとおりです。

|        | コマンド                                                                  | 目的                                                                              |
|--------|-----------------------------------------------------------------------|---------------------------------------------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# crypto ca trustpoint trustpoint-label</code> | モジュールに使用させるトラストポイントを宣言します。このコマンドをイネーブルにすると、ca-trustpoint コンフィギュレーションモードが開始されます。 |
| ステップ 2 | <code>ssl-proxy(ca-trustpoint)# crl [best-effort   optional]</code>   | CRL オプションを指定します。                                                                |

## CRL の更新

CRL は有効期限が切れるまで、以後の証明書に再利用できます。

指定された CRL の NextUpdate time に達すると、その CRL は削除されます。CRL の残り時間を表示するには、**show crypto ca timers** コマンドを入力します。

CRL が有効期間内であっても、内容が古くなっている可能性がある場合は、最新の CRL を即時ダウンロードして、古い CRL と置き換えることができます。

最新 CRL の即時ダウンロードを要求する手順は、次のとおりです。

| コマンド                                                                   | 目的                                                                           |
|------------------------------------------------------------------------|------------------------------------------------------------------------------|
| <code>ssl-proxy(config)# crypto ca crl request trustpoint_label</code> | 更新された CRL を要求します。<br><br>このコマンドを実行すると、現在保存されている CRL が最新バージョンの CRL に置き換えられます。 |



(注) 新しい CRL をダウンロードすると、CRL の既存バージョンが上書きされます。

次に、トラストポイント `tp-root` に対応する CRL をダウンロードする例を示します。

```
ssl-proxy(config)# crypto ca crl request tp-root
```

## X.500 CDP 情報の入力

CDP が X.500 DN フォーマットの場合、ホスト名とポートを入力できます。クエリで使用する情報の形式は、**ldap://hostname:[port]** です。

たとえば、検証する証明書に次の情報が含まれているとします。

- X.500 DN の設定 : **CN=CRL,O=Cisco,C=US**
- 対応するトラストポイントの設定 : **crl query ldap://10.1.1.1**

この 2 つの部分が結合されて、次のように完全な URL が形成されます。

```
ldap://10.1.1.1/CN=CRL,O=Cisco,C=US
```



(注) トラストポイントは、検証する証明書の発行元認証局の証明書と関連付ける必要があります。データベースにこのようなトラストポイントがないと、完全な URL を形成できないので、CRL のダウンロードを実行できません。

X.500 URL で CRL クエリを実行する手順は、次のとおりです。

| コマンド                                                         | 目的                                                                     |
|--------------------------------------------------------------|------------------------------------------------------------------------|
| <code>ssl-proxy(ca-trustpoint)# crl query host:[port]</code> | X.500 URL を使用して CRL クエリを実行します。CRL クエリに使用する URL は、X.500 URL でなければなりません。 |

## CRL の手動入力

認証局のサーバがオンラインで（HTTP、LDAP、または SCEP を使用して）CRL を公開しない場合、オフラインで CRL の 16 進ダンプを入手して、それを手動で入力できます。

CRL を手動で入力する手順は、次のとおりです。

|        | コマンド                                                             | 目的                                                        |
|--------|------------------------------------------------------------------|-----------------------------------------------------------|
| ステップ 1 | <code>ssl-proxy(config)# crypto ca certificate chain name</code> | 証明書チェーン コンフィギュレーション モードを開始します。 <i>name</i> で認証局の名前を指定します。 |
| ステップ 2 | <code>ssl-proxy(config-cert-chain)# crl</code>                   | 認証局が発行した失効リストを手動で入力します。                                   |
| ステップ 3 | <code>ssl-proxy(config-pubkey)# quit</code>                      | データ入力モードを終了します。                                           |
| ステップ 4 | <code>ssl-proxy(config-cert-chain)# end</code>                   | 証明書チェーン コンフィギュレーション モードを終了します。                            |

次に、CRL を手動入力する例を示します。

```
ssl-proxy(config)# crypto ca certificate chain tp
ssl-proxy(config-cert-chain)# crl
Enter the CRL in hexadecimal representation....
ssl-proxy(config-pubkey)#      30 82 01 7E 30 81 E8 30 0D 06 09 2A 86 48 86 F7
ssl-proxy(config-pubkey)#      0D 01 01 05 05 00 30 16 31 14 30 12 06 03 55 04
ssl-proxy(config-pubkey)#      03 13 0B 69 6F 73 2D 72 6F 6F 74 20 43 41 17 0D
ssl-proxy(config-pubkey)#      30 33 31 32 31 31 32 33 33 37 35 34 5A 17 0D 30
ssl-proxy(config-pubkey)#      33 31 32 31 38 32 33 33 37 35 34 5A 30 81 A0 30
ssl-proxy(config-pubkey)#      12 02 01 04 17 0D 30 33 31 30 31 34 32 32 32 35
ssl-proxy(config-pubkey)#      30 34 5A 30 12 02 01 03 17 0D 30 33 31 30 31 34
ssl-proxy(config-pubkey)#      32 32 32 35 33 30 5A 30 12 02 01 05 17 0D 30 33
ssl-proxy(config-pubkey)#      31 31 31 33 31 39 30 39 33 36 5A 30 12 02 01 06
ssl-proxy(config-pubkey)#      17 0D 30 33 31 31 31 33 31 39 32 30 34 32 5A 30
ssl-proxy(config-pubkey)#      12 02 01 07 17 0D 30 33 31 31 31 33 32 32 30 35
ssl-proxy(config-pubkey)#      35 32 5A 30 12 02 01 08 17 0D 30 33 31 31 31 33
ssl-proxy(config-pubkey)#      32 32 34 34 32 30 5A 30 12 02 01 2B 17 0D 30 33
ssl-proxy(config-pubkey)#      31 31 31 33 32 33 33 36 35 37 5A 30 12 02 01 09
ssl-proxy(config-pubkey)#      17 0D 30 33 31 31 31 33 32 33 33 37 34 38 5A 30
ssl-proxy(config-pubkey)#      0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00 03 81
ssl-proxy(config-pubkey)#      81 00 67 DE 12 99 9F C5 DF 4A F8 24 76 CE 98 4F
ssl-proxy(config-pubkey)#      7C 5C 72 1C E0 00 A9 CE 08 6E 46 8F 4D 1B FA 8E
ssl-proxy(config-pubkey)#      C9 DE CF AC 13 7D 2F BF D4 A6 C2 7B E2 31 B1 EC
ssl-proxy(config-pubkey)#      99 83 54 B3 11 24 6F C3 C3 93 C4 53 38 B6 72 86
ssl-proxy(config-pubkey)#      0A 30 F2 95 71 AE 15 66 87 3E C1 7F 8B 46 6F A9
ssl-proxy(config-pubkey)#      77 D0 FF D4 FC 73 83 79 98 BD 40 DB C1 72 9D 95
ssl-proxy(config-pubkey)#      9B 57 D1 3C 2F EF B6 63 6B 5B E4 35 40 52 2D 3A
ssl-proxy(config-pubkey)#      19 1A 4E CA 70 C6 ED 49 7A 7C 01 88 B9 CA 14 7B
ssl-proxy(config-pubkey)#      0E 1F
ssl-proxy(config-pubkey)# quit
ssl-proxy(config-cert-chain)# end
```

## CRL 情報の表示

CRL 情報を表示する手順は、次のとおりです。

| コマンド                                                | 目的                                     |
|-----------------------------------------------------|----------------------------------------|
| <code>ssl-proxy(config)# show crypto ca crls</code> | データベースに保存されている各 CRL の有効期限が切れる日時を表示します。 |

データベースに保存されている各 CRL の有効期限が切れる日時の表示例を示します。

```
ssl-proxy# show crypto ca crls
CRL Issuer Name:
  CN = test-root-CA, OU = lab, O = cisco, L = san jose, ST = california, C = US
  LastUpdate:19:08:45 UTC Dec 3 2003
  NextUpdate:20:13:45 UTC Dec 3 2003
  Retrieved from CRL Distribution Point:
    http://test-ca/CertEnroll/test-root-CA.crl
```

## CRL の削除



(注)

CRL はトラストポイント単位ではなく、グローバルに削除されます。

CRL を削除するには、トラストポイントの任意の証明書チェーンに対して、**no crl** コマンドを入力します。

CRL を削除する手順は、次のとおりです。

| コマンド                                              | 目的          |
|---------------------------------------------------|-------------|
| <code>ssl-proxy(config-cert-chain)# no crl</code> | CRL を削除します。 |

次に、CRL を削除する例を示します。

```
ssl-proxy(config)# crypto ca certificate chain tpl
ssl-proxy(config-cert-chain)# no crl
ssl-proxy(config-cert-chain)# end
```

## Certificate Security Attribute-Based Access Control

Certificate Security Attribute-Based Access Control (証明書のセキュリティ属性に基づくアクセス制御) 機能は、ACL を指定できるフィールドを証明書に追加するので、証明書に基づく ACL を作成できます。

証明書のセキュリティ属性に基づくアクセス制御の設定手順については、次の URL にアクセスして、『*Certificate Security Attribute-Based Access Control*』を参照してください。

<http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcertacl.htm>

