# OpenSSL のみを使用する証明書の生成

このセクションでは、OpenSSL を使用した Expressway の秘密キーと証明書要求の生成プロセスについて説明します。これは、フリーの OpenSSL パッケージのみに依存する一般的なプロセスで、他のソフトウェアには依存しません。これは、証明書がテスト目的でネイバーデバイスとのインターフェイスを必要とする場合や、認証局と相互作用するために出力の提供を必要とする場合に適しています。

証明書要求の生成プロセスの出力は、組織の内部または外部の認証局に提供され、Expressway がネイバーデバイスとの認証に必要とする X.509 証明書を作成するために使用できます。

ここでは、プライベート認証局の管理に OpenSSL をどのように使用できるかについても簡単に説明しますが、包括的なものではありません。これらのプロセスのさまざまなコンポーネントは、サードパーティ CA とインターフェイスするときに使用されます。

### OpenSSL および Mac OS X または Linux

OpenSSL はすでに Mac OS X にインストールされており、通常は Linux にインストールされています。

### OpenSSL と Windows

OpenSSL をまだインストールしていない場合は、http://www.openssl.org/related/binaries.html から無料でダウンロードできます。

適切な 32 ビットまたは 64 ビットの OpenSSL を選択します。「Light」バージョンで十分です。

OpenSSL のインストール中に C++ ファイルを検出できないという警告を受信した場合は、このサイトでも使用可能な“「Visual C++ 再頒布可能パッケージ」”をロードし、OpenSSL ソフトウェアをリロードします。

この章では、次の内容について説明します。

# OpenSSL を使用する証明書要求の作成

このプロセスでは、後で CA が検証する可能性があるサーバーの秘密キーと証明書要求が作成されます。これは、ローカルで作成および管理されている CA やサードパーティ CA にすることができます。

（注）
- 証明書署名要求を作成するこの方法は、コマンドが誤って入力される可能性があるため（特にSANエントリが多数ある場合）、OpenSSLでの作業に関する詳しい知識を持っている場合にのみ使用してください。関連するSANエントリが不足していると、証明書を後日再作成する必要があります。
- バージョン X8.5.1 から、ユーザーインターフェイスにダイジェストアルゴリズムを設定するオプションがあります。デフォルトでは SHA-256 に設定されており、SHA-1、SHA-384、または SHA-512 に変更するオプションがあります。

OpenSSL のコマンドラインから証明書署名要求を生成するには、次の手順を使用します。

### 手順

**Step 1** Expressway に SSH 接続し、root としてログインします。

**Step 2** 動作する新しいディレクトリを作成します：`mkdir /tmp/certtemp`

**Step 3** ディレクトリを移動します：`cd /tmp/certtemp`

**Step 4** 編集する必要があるため、証明書署名要求に使用するOpenSSL構成ファイルをこのディレクトリにコピーします（**注：コマンドの末尾のドットもそのまま付けます**）：`cp /etc/openssl/csrreq.cnf`

**Step 5** 編集するためにファイルを開きます：`vi csrreq.cnf`

**Step 6** 「`default_md = sha1`」 の行を検索し、その行が 「`default_md = sha256`」 となるように編集します。

**Step 7** 「`# req_extensions = v3_req`」 行の先頭の **#** を削除して、コメントを解除します。

**Step 8** 「`extendedKeyUsage=serverAuth, clientAuth`」 行が、`[v3_req]` セクションで表示されていることを確認します。

**Step 9** 「`subjectAltName = ${ENV::CSR_ALT_NAME}`」 行を検索し、証明書内のサブジェクト代替名に希望するものがリストされるように置き換えます。例えば、「`subjectAltName = DNS:peer1vcs.example.com,DNS:peer2vcs.example.com,DNS:ClusterFQDN.example.com`」 にようにします。関連するすべてのエントリを追加したことを確認します。MRAの場合、次のように構成されます。

1. Expressway E: `DNS:<CM domain name>, DNS:<XMPP federation domain>, DNS:<federation chat alias 1>, DNS:<federation chat alias 2>`、など。
2. Expressway C: `DNS:<secure profile name 1>, DNS:<secure profile name 2>`、など。

OpenSSL のみを使用する証明書の生成

**Step 10** ファイルを保存して終了します。

**Step 11** 次の OpenSSL コマンドを実行して、必要に応じて、VCS 「`openssl req -nodes -newkey rsa:4096 -keyout privatekey.pem -out myrequest.csr -config csrreq.cnf`」 changing the `rsa:nnnn` 用に新しい証明書署名要求と秘密キーを生成します。（nnnn = キー長、推奨値は 4096）。

**Step 12** 情報を入力する必要がある次の例のような出力がコンソールに表示されます。すべてを入力する必要はありませんが、一部のフィールドは必須です。

- 国
- 都道府県
- 地域の名前
- 組織名
- 共通名
- 電子メール アドレス：任意、空欄のままでも可
- チャレンジパスワード：任意、空欄のままでも可
- 任意の会社名：任意、空欄のままでも可

```
Generating a 4096 bit RSA private key
.............++
.................................................++
writing new private key to 'privatekey.pem'
-----
証明書要求に記載する情報を入力するように求められています。
ここで入力するのは、識別名または DN と呼ばれる情報です。
入力するフィールドはごく限られており、一部は空白のままにすることもできます。
デフォルト値が入っているフィールドもあります。
「.」を入力すると空白のままになります。
-----
国名（2 文字コード） [AU]:GB
都道府県（フルネーム）[Some-State]:Berkshire
地域の名前（市など） []:Reading
組織名（例：会社名）[Internet Widgits Pty Ltd]:Cisco
組織の部署名（例：部門）[]:CIBU
共通名（例：自分の名前）[]:exp01.example.com
電子メールアドレス []:
```

フィールドに入力すると、**myrequest.csr** と **privatekey.pem** の 2 つの新しいファイルがあります。

**Step 13** （オプション）ドメインネームシステム（DNS）エントリが正常に要求に入力されているかを確認する場合は、`openssl req -text -noout -in myrequest.csr` コマンドを使用して、**myrequest.csr** ファイルを復号化します。

**Step 14** 証明書署名要求を選択した認証局に送信します。その認証局からは公開証明書が提供されます。

**Step 15** **[メンテナンス（Maintenance）] > [セキュリティ（Security）] > [サーバー証明書（Server certificate）]** Web ページの順に選択し、「**[サーバー証明書ファイルの選択（Select the server certificate file）]**」 入力ボックスから、公開証明書を VCS にアップロードします。

**Step 16** **[メンテナンス（Maintenance）] > [セキュリティ（Security）] > [サーバー証明書（Server certificate）]** Web ページの順に選択し、「**[サーバー秘密キーファイルの選択（Select the server private key file）]**」 入力ボックスから、**privatekey.pem** を VCS にアップロードします。

---

**privatekey.pem** を安全な場所で保管します。

# OpenSSL を使用する認証局としての操作

主要な展開では、サードパーティの認証局を使用するか、または組織の IT 部門にすでに内部認証局が 1 つ存在する可能性があります。ただし、次に説明するように、OpenSSL を使用してプライベート認証局で証明書を管理することができます。

CA として機能するようにすでに OpenSSL を構成している場合は、OpenSSL を使用する署名付き証明書の作成項に進みます。

## OpenSSL を CA として構成する

OpenSSL は強力なソフトウェアで、CA として動作するには、発行された証明書を追跡するためのいくつかのディレクトリとデータベースの設定が必要です。

ディレクトリとファイルのリストは、OpenSSL 構成ファイルの `[CA_default]` セクションで確認できます。デフォルトでは、必要なファイル/ディレクトリを作成します。

- **certs**、**newcerts** および **private** の 3 つのサブディレクトリがある、現在のディレクトリ内の **demoCA** ディレクトリ。
- **demoCA** ディレクトリ内にある **index.txt** という空のファイル。
- 2 桁の番号（「10」 など）を保存している **demoCA** ディレクトリ内の **serial** というファイル。

たとえば、次のコマンドを使用します。

```
mkdir demoCA

cd demoCA

mkdir certs

mkdir newcerts
```

```
mkdir private

touch index.txt

echo 10 > serial
```

# OpenSSL を使用する認証局の作成

このプロセスで、認証局（CA）の秘密キーと証明書が作成され、他の証明書を検証するために使用可能になります。これは明示的にインストールされるもの以外のデバイスから信頼されることはないことに注意してください。

コマンドプロンプトから次を実行します。

### 手順

**Step 1** **demoCA** ディレクトリにいることを確認します。

**Step 2** Windows の場合：OpenSSL が **demoCA** ディレクトリにインストールされているディレクトリから **openssl.cfg** をコピーし、その名前を **openssl_local.cfg** に変更します。

Mac OS X の場合：**/System/Library/OpenSSL/openssl.cnf** を **demoCA** ディレクトリにコピーし、名前を **openssl_local.cfg** に変更します。

**Step 3** テキストエディタを使用して、上記のコピーコマンドで作成された **openssl_local.cfg** ファイルを編集します。`[CA_default]` セクションに次の修正を行います。

1. `copy_extensions = copy` 行の先頭に # が無いことを確認します。# がある場合は、削除します。行がコメントアウトされたままの場合は、証明書署名要求の属性が除去され、SSL サーバーと SSL クライアントの属性は証明書に表示されません。
2. `policy = policy_match` から `policy = policy_anything` に変更します。
3. `dir = ./demoCA` to `dir =` を変更します。
4. 任意で `default_days = 365`（生成された証明書の効力が 1 年）を `default_days = 3650`（10 年、または適切な値を選択）に変更します。
5. ファイルを保存します。

**Step 4** 次のコマンドを実行して、CA の秘密キーを生成します。

```
openssl genrsa -aes256 -out private/cakey.pem 4096
```

ここで、秘密キーを暗号化するパスワードが求められるので、強力なパスワードを選択し、安全な場所に記録します。cakey.pem ファイルが CA 証明書を作成し、他の証明書に署名するために使用されるので、安全に保持する必要があります。

**Step 5** 次のコマンドを実行して、CA 証明書を生成します。

Windows の場合：`openssl req -new -x509 -days 3650 -key private/cakey.pem -config openssl_local.cfg -sha1 -extensions v3_ca -out cacert.pem`

OS X の場合：`openssl req -new -x509 -days 3650 -key private/cakey.pem -config openssl_local.cfg -sha1 -extensions v3_ca -out cacert.pem`

**Step 6** キーのパスフレーズを入力し、次の項目を含む要求されたデータを入力します。

- 国
- 都道府県
- 地域の名前
- 組織名
- 組織単位
- 共通名：通常は、この CA の担当者の名前になります
- 電子メールアドレス：任意、空欄のままでも可

要求されたデータを入力すると、処理が完了し、認証局の証明書 **cacert.pem** が使用可能になります。

# OpenSSL を使用する署名付き証明書の作成

このプロセスでは、以前に生成された証明書要求を使用して生成された CA キーでサーバー証明書に署名します。

コマンドプロンプトから次を実行します。

### 手順

**Step 1** **demoCA** ディレクトリにいることを確認します。

**Step 2** 証明書要求ファイル（**certcsr.pem**）が使用できることを確認します。

- Expressway を使用して証明書要求を作成する場合は、次の手順を実行します（推奨プロセス）。

  Expressway からダウンロードしたファイルを **demoCA** ディレクトリにコピーし、名前を **certcsr.pem** に変更します。

- OpenSSL を使用して証明書要求を作成する場合は、次の手順を実行します。

  以前に生成された証明書要求を **demoCA** ディレクトリにコピーして、次のコマンドを実行して PEM フォーマットに変換します。

  ```
  openssl req -in certcsr.der -inform DER -out certcsr.pem -outform PEM
  ```

**Step 3** 次のコマンドを実行して署名済みサーバー証明書を生成します。

```
openssl ca -config openssl_local.cfg -cert cacert.pem -keyfile private/cakey.pem -in
certcsr.pem -out certs/server.pem -md sha1
```

「failed to update database TXT_DB error number 2」というエラーメッセージを受信した場合、index.txt ファイルの内容を削除してからコマンドを再実行します。

**Step 4** CA の秘密キーのパスワードを入力するよう求められます。

---

サーバー用の署名済み証明書が **demoCA/certs/server.pem** として使用可能になります。

# OpenSSL OpenSSL を使用する自己署名付き証明書の作成

自己署名証明書を作成することは推奨しません。Unified Communications 展開では動作しません。

その代わりに、前述のように OpenSSL を使用して認証局を作成する必要があります。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。