

IOS XE証明書を生成するためのOpenSSLでのマルチレベルCAの設定

内容

[はじめに](#)

[前提条件](#)

[要件](#)

[使用するコンポーネント](#)

[設定](#)

[概要](#)

[OpenSSL設定ファイルの準備](#)

[認証局の初期ファイルの作成](#)

[ルートCA証明書の作成](#)

[中間CA証明書の作成](#)

[デバイス証明書の作成](#)

[Cisco IOS XEデバイス証明書の作成](#)

[オプション：エンドポイント証明書の作成](#)

[Cisco IOS XEデバイスへの証明書のインポート](#)

[確認](#)

[OpenSSLでの証明書情報の確認](#)

[トラブルシューティング](#)

[失効チェックが実行されている](#)

[関連情報](#)

はじめに

このドキュメントでは、マルチレベルCAを作成して、Cisco IOS® XEデバイスと互換性のある汎用証明書を作成する方法について説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- OpenSSL アプリケーションを使用する方法。
- 公開キーインフラストラクチャ(PKI)とデジタル証明書。

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

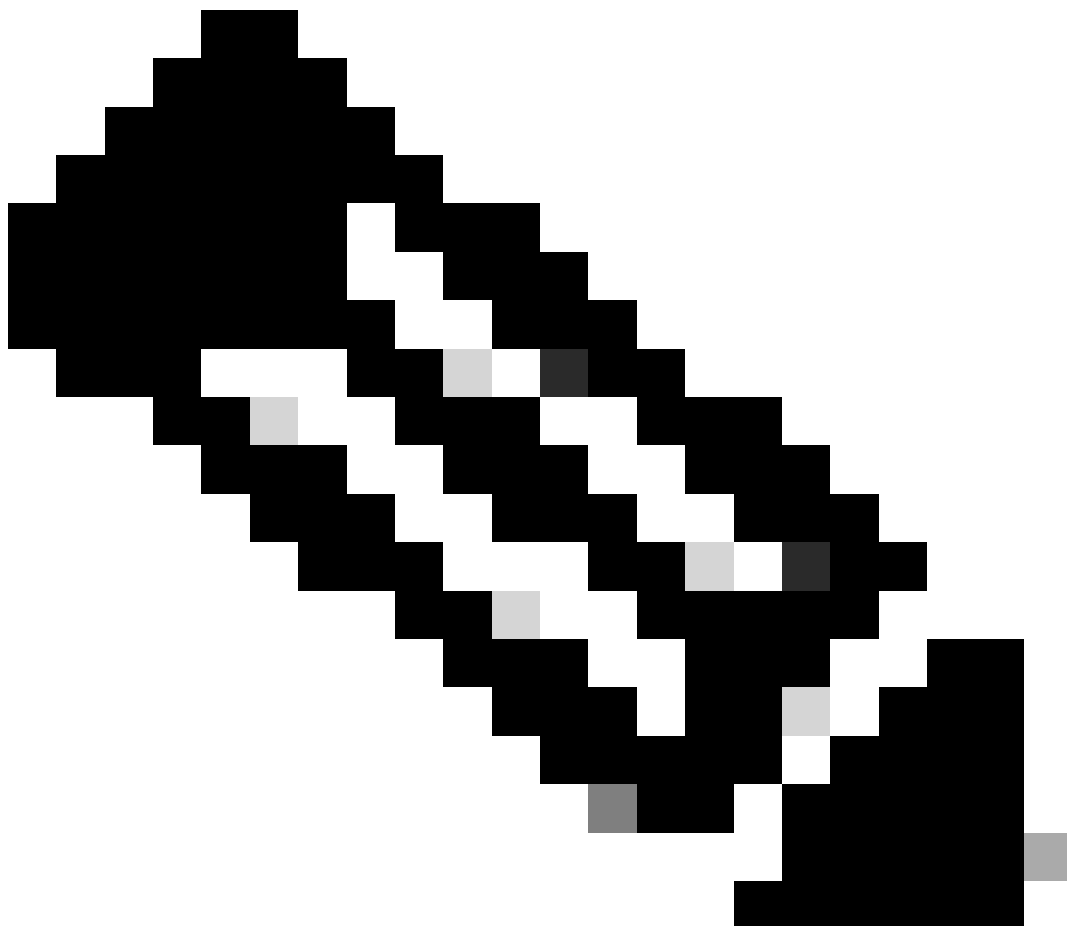
- OpenSSLアプリケーション (バージョン3.0.2)
- 9800 WLC (Cisco IOS XEバージョン17.12.3)。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな (デフォルト) 設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

設定

概要

この目的は、デバイス証明書に署名するためのルートCAと中間CAを持つ2つのレベルのローカル認証局(CA)を作成することです。証明書が署名されると、Cisco IOS XEデバイスにインポートされます。



注：このドキュメントでは、Linux固有のコマンドを使用してファイルを作成および配置

します。OpenSSLを使用できる他のオペレーティングシステムでも同じ操作を実行できるように、コマンドについて説明します。

OpenSSL設定ファイルの準備

OpenSSLがインストールされているマシンの現在の作業ディレクトリからopenssl.confというテキストファイルを作成します。これらの行をコピーアンドペーストして、証明書署名に必要な設定をOpenSSLに提供します。必要に応じてこのファイルを編集できます。

```
[ ca ]
default_ca = IntermCA

[ RootCA ]

dir      = ./RootCA
certs    = $dir/RootCA.db.certs
crl_dir  = $dir/RootCA.db.crl
database = $dir/RootCA.db.index
unique_subject = yes
new_certs_dir = $dir/RootCA.db.certs
certificate = $dir/RootCA.crt
serial    = $dir/RootCA.db.serial
#crlnumber = $dir/RootCA.db.crlserial
private_key = $dir/RootCA.key
RANDFILE  = $dir/RootCA.db.rand
name_opt  = ca_default
cert_opt  = ca_default
##### Modify default days for certificates signed by Root CA (Intermediate cert)
default_days = 360
default_md   = sha256
preserve     = no
policy       = optional_policy

[ IntermCA ]

dir      = ./IntermCA
certs    = $dir/IntermCA.db.certs
crl_dir  = $dir/IntermCA.db.crl
database = $dir/IntermCA.db.index
unique_subject = yes
new_certs_dir = $dir/IntermCA.db.certs
certificate = $dir/IntermCA.crt
serial    = $dir/IntermCA.db.serial
private_key = $dir/IntermCA.key
RANDFILE  = $dir/IntermCA.db.rand
name_opt  = ca_default
cert_opt  = ca_default
# Certificate field options
##### Modify default days for certificates signed by Intermediate CA cert (devi
default_days = 1000
#default_crl_days = 1000
default_md   = sha256
# use public key default MD
preserve     = no
policy       = optional_policy
```

```
[ optional_policy ]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName  = optional
organizationalUnitName = optional
commonName       = supplied
```

```
[ req ]
default_bits      = 2048
default_keyfile   = privkey.pem
distinguished_name = req_distinguished_name
attributes        = req_attributes
x509_extensions  = v3_ca # The extensions to add to the signed cert
string_mask       = nombstr
```

```
[ req_distinguished_name ]
countryName          = Country Name
countryName_default  = MX
countryName_min      = 2
countryName_max      = 2

stateOrProvinceName = State or province
stateOrProvinceName_default = CDMX
```

```
localityName         = Locality
localityName_default = CDMX
```

```
organizationName     = Organization name
organizationName_default = Cisco lab
```

```
organizationalUnitName = Organizational unit
organizationalUnitName_default = Cisco Wireless
```

```
commonName           = Common name
commonName_max       = 64
```

```
[ req_attributes ]
# challengePassword = A challenge password
# challengePassword_min = 4
# challengePassword_max = 20
```

#This section contains the extensions used for the Intermediate CA certificate

```
[ v3_ca ]
# Extensions for a typical CA
basicConstraints = CA:true
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
subjectAltName = @Intermediate_alt_names
```

```
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
```

```

[ crl_ext ]
# CRL extensions.
#authorityKeyIdentifier=keyid:always,issuer:always

#DEFINE HERE SANS/IPs NEEDED for Intermediate CA device certificates
[Intermediate_alt_names]
DNS.1 = Intermediate.example.com
DNS.2 = Intermediate2.example.com

#Section for endpoint certificate CSR generation
[ endpoint_req_ext ]
subjectAltName = _alt_names

#Section for endpoint certificate sign by CA
[ Endpoint ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#Change the key usage according to the certificate usage needs
extendedKeyUsage = clientAuth
subjectAltName = _alt_names

#Define here SANS/IPs needed for Endpoint certificates
[endpoint_alt_names]
DNS.1 = Endpoint.example.com
DNS.2 = Endpoint2.example.com

#Section for IOS-XE device certificate CSR generation
[ device_req_ext ]
subjectAltName = @IOS_alt_names

#Section for IOS-XE certificate sign by CA
[ IOS_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#Change the key usage according to the certificate usage needs
extendedKeyUsage = clientAuth , serverAuth
subjectAltName = @IOS_alt_names

#Define here SANS/IPs needed for IOS-XE certificates
[IOS_alt_names]
DNS.1 = IOSXE.example.com
DNS.2 = IOSXE2.example.com

```

認証局の初期ファイルの作成

現在のディレクトリにRootCAという名前のフォルダを作成します。その中に、RootCA.db.tmp、RootCA.db.certs、およびRootCA.db.crlという3つのフォルダを作成します。

```

mkdir RootCA
mkdir RootCA/RootCA.db.tmp
mkdir RootCA/RootCA.db.certs
mkdir RootCA/RootCA.db.crl

```

RootCAフォルダ内にRootCA.db.serialというファイルを作成します。このファイルには、証明書のシリアル番号の初期値を含める必要があります。この例では、01が選択された値です。

RootCAフォルダ内にRootCA.db.crlserialというファイルを作成します。このファイルには、証明書失効リスト(CRL)番号の初期値が含まれている必要があります。この場合は、01が選択されています。

```
echo 01 > RootCA/RootCA.db.serial  
echo 01 > RootCA/RootCA.db.crlserial
```

RootCAフォルダ内にRootCA.db.indexというファイルを作成します。

```
touch RootCA/RootCA.db.index
```

RootCAフォルダ内にRootCA.db.randという名前のファイルを作成し、内部の乱数生成器のシードとして機能するように、このファイルに8192のランダムバイトを設定します。

```
openssl rand -out RootCA/RootCA.db.rand 8192
```

現在のディレクトリにIntermCAというフォルダを作成します。その中に、IntermCA.db.tmp、IntermCA.db.certs、およびIntermCA.db.crlという3つのフォルダを作成します。

```
mkdir IntermCA  
mkdir IntermCA/IntermCA.db.tmp  
mkdir IntermCA/IntermCA.db.certs  
mkdir IntermCA/IntermCA.db.crl
```

IntermCAフォルダ内にIntermCA.db.serialというファイルを作成します。このファイルには、証明書のシリアル番号の初期値を含める必要があります。この例では、01が選択された値です。

IntermCAフォルダ内にIntermCA.db.crlserialというファイルを作成します。このファイルには、証明書失効リスト(CRL)番号の初期値が含まれている必要があります。この場合は、01が選択されています。

```
echo 01 > IntermCA/IntermCA.db.serial  
echo 01 > IntermCA/IntermCA.db.crlserial
```

IntermCAフォルダ内にIntermCA.db.indexという名前のファイルを作成します。

IntermCAフォルダ内にIntermCA.db.randという名前のファイルを作成し、内部の乱数生成器のシードとして機能するように8192のランダムバイトを設定します。

```
touch IntermCA/IntermCA.db.index
```

IntermCAフォルダ内にIntermCA.db.randという名前のファイルを作成し、内部の乱数生成器のシードとして機能するように8192のランダムバイトを設定します。

```
openssl rand -out IntermCA/IntermCA.db.rand 8192
```

これは、すべての初期ルートおよび中間CAファイルを作成した後のファイル構造です。

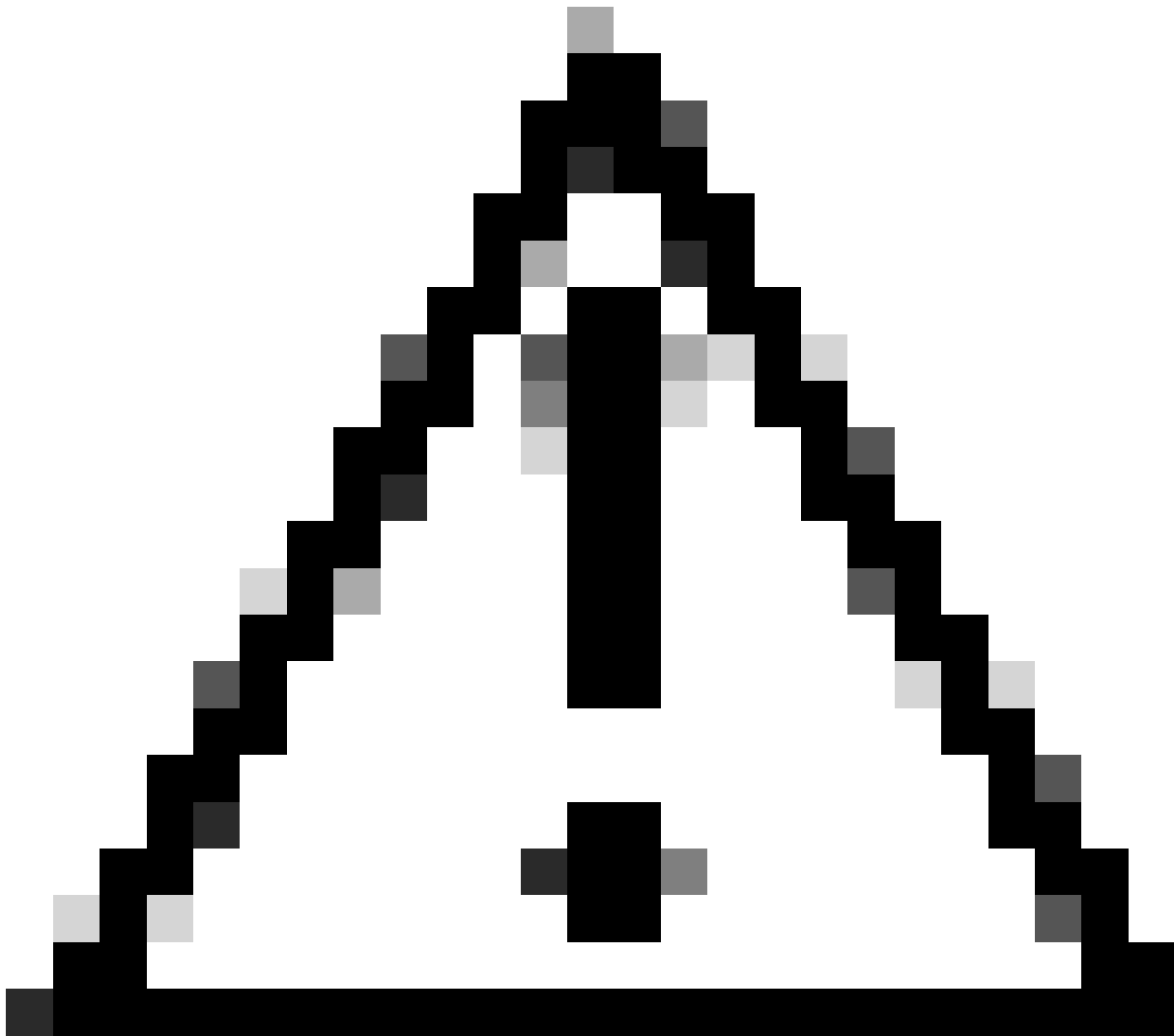
```
mariomed@CSC0-W-PF320YP6:/mnt/c/Users/mariomed/radsecfiles1$ tree
```

```
├── IntermCA
│   ├── IntermCA.db.certs
│   ├── IntermCA.db.crl
│   ├── IntermCA.db.crlserial
│   ├── IntermCA.db.index
│   ├── IntermCA.db.rand
│   ├── IntermCA.db.serial
│   └── IntermCA.db.tmp
├── RootCA
│   ├── RootCA.db.certs
│   ├── RootCA.db.crl
│   ├── RootCA.db.crlserial
│   ├── RootCA.db.index
│   ├── RootCA.db.rand
│   ├── RootCA.db.serial
│   └── RootCA.db.tmp
└── openssl.cnf
```

ルートCA証明書の作成

次のコマンドを実行して、ルートCAの秘密キーを作成します。

```
openssl genrsa -des3 -out ./RootCA/RootCA.key 4096
```



注意: OpenSSLでは、キーを生成する際にパスワードの入力が必要です。パスワードのシークレットと、生成された秘密キーを安全な場所に保管します。アクセス権を持つすべてのユーザが、ルートCAとして証明書を発行できます。

opensslでreqコマンドを使用して、ルートCA自己署名証明書を作成します。-x509フラグは、証明書署名要求(CSR)を内部で作成し、自動的に自己署名します。-daysのパラメータとサブジェクト代替名を編集します。共通の名前を入力するようプロンプトが表示されます。入力する共通名(CN)がサブジェクト代替名(SAN)と一致していることを確認します。

```
openssl req -new -key ./RootCA/RootCA.key -out ./RootCA/RootCA.crt -config openssl.cnf -x509 -days 3650
```



```
karlowed@CSCO-W-PF328YP6:~$ openssl req -new -x509 -days 3650 -key ./RootCA/RootCA.key -out ./RootCA/RootCA.crt -config openssl.cnf
Enter pass phrase for ./RootCA/RootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name [MX]:
State or province [CDMX]:
Locality [CDMX]:
Organization name [Cisco Lab]:
Organizational unit [Cisco Wireless]:
Common name [ ]; Wireless TAC Root
Email Address [ ]:
```

OpenSSL Distinguished Nameインタラクティブプロンプト

生成されたファイルはRootCA.crtという名前で、RootCAフォルダ内にあります。このファイルはルートCA証明書です。

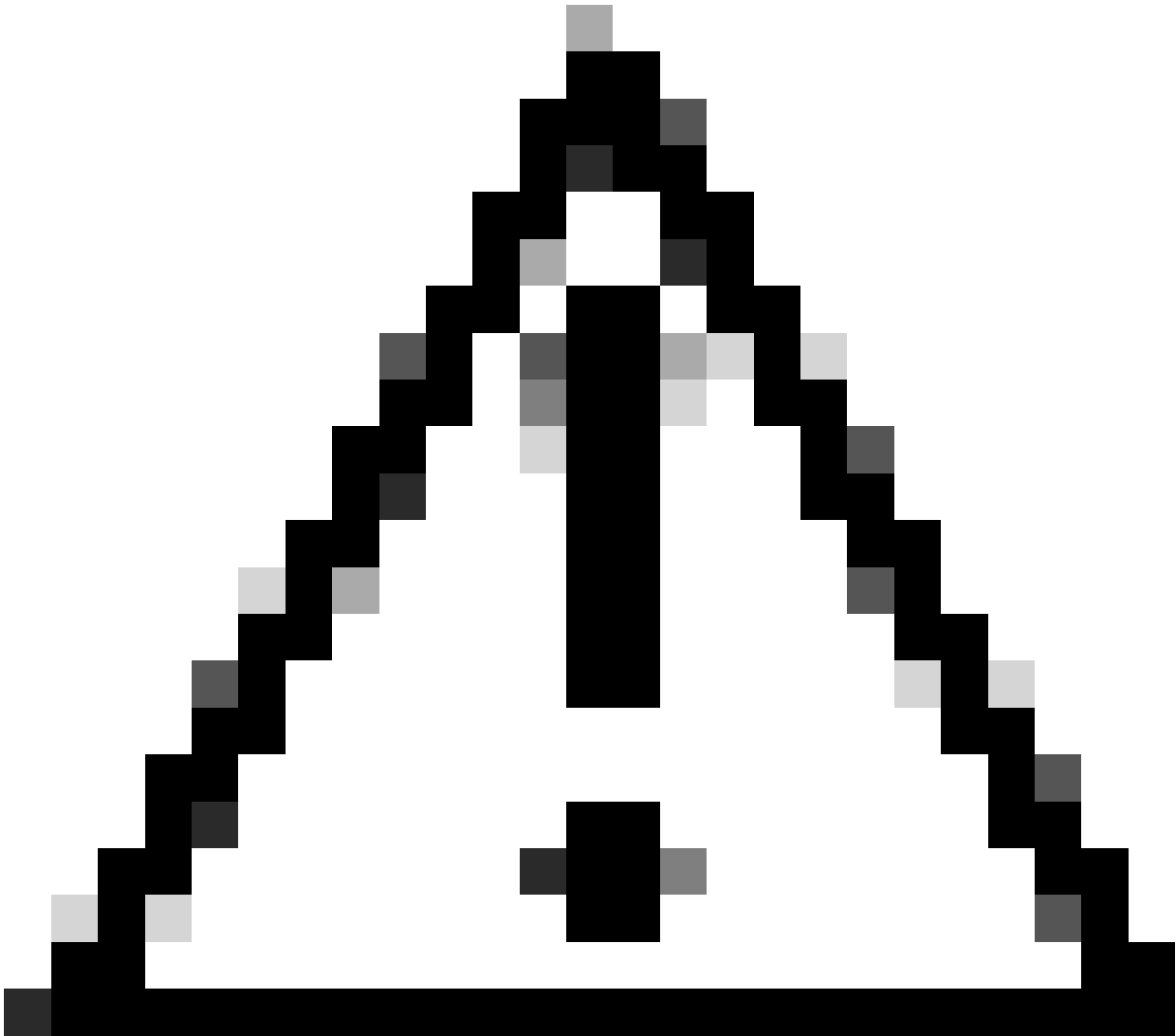
中間CA証明書の作成

ルートフォルダ内に署名済み中間CA証明書を格納するフォルダを作成します。

```
mkdir ./RootCA/RootCA.db.certs/IntermCA
```

中間証明書用の秘密キーを作成します。

```
openssl genrsa -des3 -out ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key 4096
```



注意: OpenSSLでは、キーを生成する際にパスワードの入力が必要です。パスワードのシークレットと、生成された秘密キーを安全な場所に保管します。このCAにアクセスできるユーザは、中間CAとして証明書を発行できます。

中間CA証明書署名要求を作成します。端末から、証明書情報の入力を求めるプロンプトが表示されます。

```
openssl req -new -key ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key -out ./RootCA/RootCA.db.certs/IntermCA/IntermCA.req
```

openssl.cnfファイルのRootCAセクションを使用して、中間CSRに署名します。

```
openssl ca -config openssl.cnf -name RootCA -extensions v3_ca -out ./RootCA/RootCA.db.certs/IntermCA/IntermCA.csr
```

生成されたファイルはIntermCA.crtという名前で、RootCAフォルダ内にあります。このファイルはルートCA証明書です。

中間CAの初期ファイルの一部として作成した独自のフォルダに、中間の証明書とキーを移動します。

```
cp ./RootCA/RootCA.db.certs/IntermCA/IntermCA.crt ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key ./Inte
```

これは、最初のルートCAと中間CAの両方の秘密キーと証明書を作成した後のファイル構造です。

```
mariomed@CSCO-W-PF320YP6:/mnt/c/Users/mariomed/radsecfiles$ tree
```

```
.
├── IntermCA
│   ├── IntermCA.crt <-----Intermediate CA certficate
│   ├── IntermCA.db.certs
│   ├── IntermCA.db.crl
│   ├── IntermCA.db.crlserial
│   ├── IntermCA.db.index
│   ├── IntermCA.db.rand
│   ├── IntermCA.db.serial
│   ├── IntermCA.db.tmp
│   └── IntermCA.key <-----Intermediate CA private key
├── RootCA
│   ├── RootCA.crt <-----Root CA certficate
│   ├── RootCA.db.certs
│   │   ├── 01.pem
│   │   └── IntermCA
│   │       ├── IntermCA.crt
│   │       ├── IntermCA.csr
│   │       └── IntermCA.key
│   ├── RootCA.db.crl
│   ├── RootCA.db.crlserial
│   ├── RootCA.db.index
│   ├── RootCA.db.index.attr
│   ├── RootCA.db.index.old
│   ├── RootCA.db.rand
│   ├── RootCA.db.serial
│   ├── RootCA.db.serial.old
│   ├── RootCA.db.tmp
│   └── RootCA.key <-----Root CA private key
└── openssl.cnf
```

デバイス証明書の作成

Cisco IOS XEデバイス証明書の作成

Cisco IOS XEデバイス証明書を保存する新しいフォルダを作成します。

```
mkdir ./IntermCA/IntermCA.db.certs/IOSdevice
```

デバイス秘密キーIOSdevice.keyとデバイスCSR IOSdevice.csrを作成します。device_req_extセクションを使用して、そのセクションのSANをCSRに追加します。

```
openssl req -newkey rsa:4096 -sha256 -keyout ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdevice.key -node
```

CSRに指定する共通名がSANに一致するように、openssl.cnfファイルの[IOS_alt_names]セクションを変更します。

```
#Define here SANS/IPs needed for IOS-XE certificates
[IOS_alt_names]
DNS.1   = IOSXE.example.com
DNS.2   = IOSXE2.example.com
```

中間CA IntermCAセクションでIOS XEデバイスCSRに署名します。openSSLコンフィギュレーションファイルをポイントするには-configを、IOS_certセクションをポイントするには-extensionsを使用します。これにより、署名付き証明書にSANが保持されます。

```
openssl ca -config openssl.cnf -extensions IOS_cert -name IntermCA -out ./IntermCA/IntermCA.db.certs/IO
```

この手順の後、IOS XEデバイス用の有効な証明書IOSdevice.crtを作成し、一致する秘密キーIOSdevice.keyを指定しました。

オプション：エンドポイント証明書の作成

この時点で、ローカルCAを導入し、IOS XEデバイスに対して1つの証明書を発行しました。このCAを使用して、エンドポイントID証明書を生成することもできます。これらの証明書は、9800ワイヤレスLANコントローラでのローカルEAP認証や、RADIUSサーバでのdot1x認証などを実行する場合にも有効です。このセクションでは、エンドポイント証明書を生成できます。

エンドポイント証明書を保存するフォルダを作成します。

```
mkdir ./IntermCA/IntermCA.db.certs/Endpoint
```

CSRに指定する共通名がSANと一致するように、openssl.cnfファイル[endpoint_alt_names]セクションを変更します。

```
#Define here SANS/IPs needed for Endpoint certificates
[endpoint_alt_names]
DNS.1 = Endpoint.example.com
DNS.2 = Endpoint2.example.com
```

セクションendpoint_req_ext for SANsを使用して、エンドポイントの秘密キーとWLC CSRを作成します。

```
openssl req -newkey rsa:2048 -keyout ./IntermCA/IntermCA.db.certs/Endpoint/Endpoint.key -nodes -config
```

エンドポイントデバイス証明書に署名します。

```
openssl ca -config openssl.cnf -extensions Endpoint -name IntermCA -out ./IntermCA/IntermCA.db.certs/En
```

Cisco IOS XEデバイスへの証明書のインポート

ルートCAと中間CAを含むファイルを同じファイルに作成し、Cisco IOS XEデバイスへのインポートに必要なcertfile.crtという名前を付けて./IntermCA/IntermCA.db.certs/WLC/フォルダに保存します。

```
cat ./RootCA/RootCA.crt ./IntermCA/IntermCA.crt > ./IntermCA/IntermCA.db.certs/IOSdevice/certfile.crt
```

9800シリーズのWLCでは、証明書インポート用のpfxファイルの作成に、さまざまなコマンドを使用します。pfxファイルを作成するには、Cisco IOS XEバージョンに応じて、次のコマンドのいずれかを実行します。

証明書インポートプロセスの詳細については、『[Catalyst 9800 WLCでのCSR証明書の生成とダウンロード](#)』を参照してください

17.12.1より前のバージョンの場合 :

```
openssl pkcs12 -export -macalg sha1 -legacy -descert -out ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdev
```

バージョン17.12.1以降 :

```
openssl pkcs12 -export -out ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdevice.pfx -inkey ./IntermCA/Inte
```

Cisco IOS XEデバイスにIOSdevice.pfx証明書をインポートします。

```
WLC# configure terminal
WLC(config)#crypto pki import
```

```
pkcs12 [tftp://
```

```
/
```

```
| ftp://
```

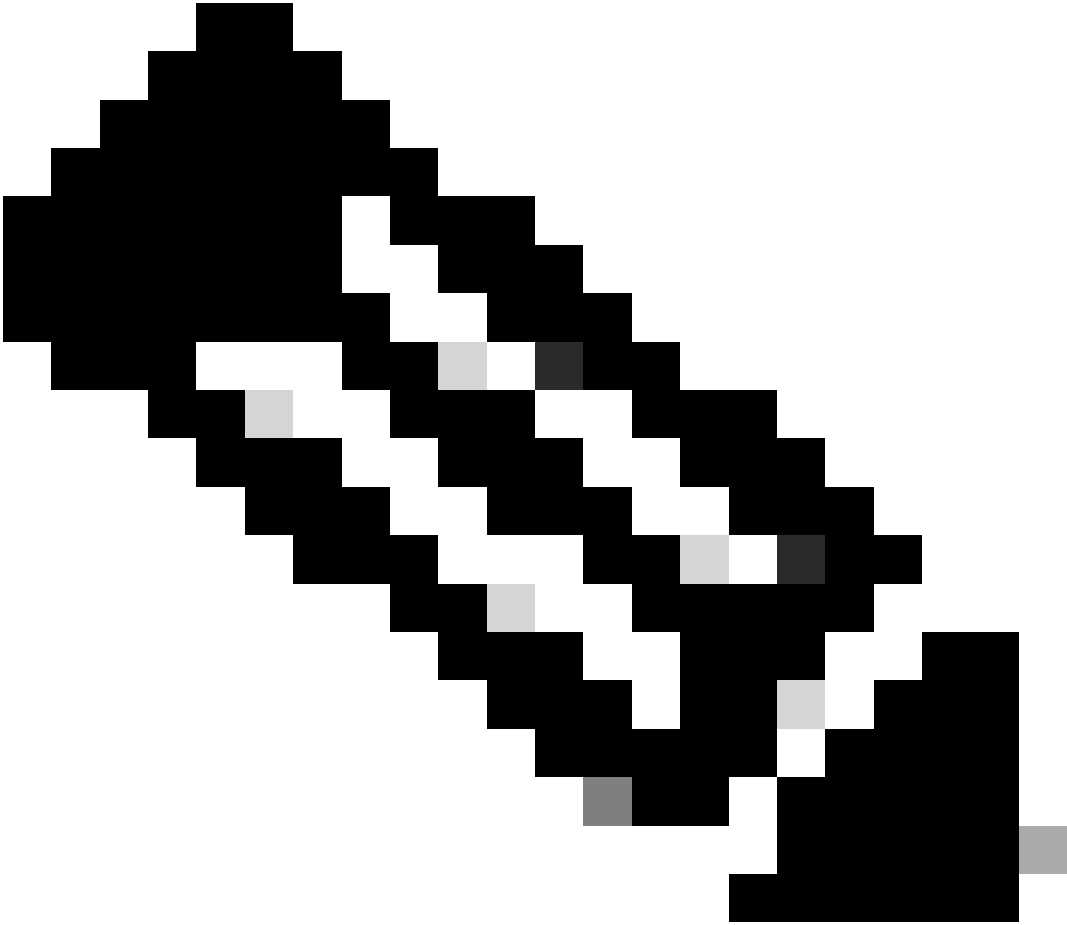
```
/
```

```
| http://
```

/

| bootflash:

] password



注：このガイド用に作成したCA証明書が、デバイス証明書を検証する必要があるデバイスによって信頼されていることを確認してください。たとえば、Cisco IOS XEデバイスでデバイス証明書がWeb管理の目的で使用される場合、管理ポータルにアクセスするすべてのコンピュータまたはブラウザで、信頼ストアにCA証明書が必要です。

Cisco IOS XEデバイスが展開したCAから確認できるオンライン証明書失効リストがないため、証明書の失効チェックを無効にします。

検証パスの一部であるすべてのトラストポイントで無効にする必要があります。ルートCAトラストポイントは、文字列-rrr1が末尾に追加された中間/デバイストラストポイントと同じ名前です。

```
9800#configure terminal
```

```
9800(config)#crypto pki trustpoint IOSdevice.pfx
```

```
9800(config)#revocation-check none
```

```
9800(config)#exit
```

```
9800(config)#crypto pki trustpoint IOSdevice.pfx-rrr1
```

```
9800(config)#revocation-check none
```



```
9800(config)#exit
```

確認

OpenSSLでの証明書情報の確認

作成した証明書の証明書情報を確認するには、Linuxターミナルで次のコマンドを実行します。

```
openssl x509 -in
```

```
-text -noout
```

完全な証明書情報が表示されます。

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2 (0x2)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = MX, ST = CDMX, L = CDMX, O = Cisco lab, OU = Cisco Wireless, CN = Intermediate.example.com
    Validity
      Not Before: Jul 18 19:14:57 2024 GMT
      Not After : Apr 14 19:14:57 2027 GMT
    Subject: C = MX, ST = CDMX, L = CDMX, O = Cisco lab, OU = Cisco Wireless, CN = WLC.example.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:b1:10:7d:6c:6c:14:2f:18:a6:0b:69:d9:60:03:
        56:2d:48:22:f0:42:10:65:44:24:3b:54:e1:4b:87:
        b8:ab:c5:5f:f6:a1:a3:5e:f6:3c:c5:45:cc:01:6d:
        df:e8:a7:81:28:50:44:54:4c:af:a0:56:cf:06:be:
        10:7e:e2:46:42:ea:3c:b9:d4:03:75:08:84:70:36:
        bb:3d:95:3b:e2:86:e6:f7:d9:4d:00:28:c4:3c:cb:
        f8:6d:37:5c:89:28:c1:75:b1:7e:fa:bd:91:cf:8e:
        5c:a2:37:4f:71:da:6a:04:ee:ba:68:bf:4d:f2:d3:
        ae:aa:13:42:3b:ff:a0:b3:65:c9:ff:f6:9a:06:d7:
        6c:08:10:e0:b9:d8:ca:93:2d:e5:5d:7b:74:cd:93:
        68:b1:46:c7:35:d7:6b:0f:a6:ae:34:e6:23:d1:c8:
        d3:bf:c0:85:ab:2d:02:a8:dd:54:77:e3:32:61:4e:
        33:58:b0:62:12:82:42:ae:2b:69:f0:5f:0c:90:c7:
        9c:ef:b9:9c:fc:29:e2:2c:cb:b4:a9:01:fa:5d:3c:
        97:11:67:cc:25:96:01:3d:26:1a:43:34:bd:43:b0:
        a0:f1:ec:a0:c7:98:ad:32:32:99:9c:6b:61:af:57:
        53:ee:20:cc:d5:ed:db:1c:5c:65:51:42:8c:28:bf:
        62:bf
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Subject Key Identifier:
        87:89:CA:28:06:95:D5:CE:7C:66:B4:75:81:AA:D4:19:EC:43:01:BB
      X509v3 Authority Key Identifier:
        keyid:2B:08:D8:4C:23:72:5B:62:03:EA:44:F6:9E:D9:F7:75:2E:64:97:DE
        DirName:/C=MX/ST=CDMX/L=CDMX/O=Cisco lab/OU=Cisco Wireless/CN=RootCA
        serial:01
      X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 Subject Alternative Name:
        DNS:WLC.example.com, DNS:WLC2.example.com
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:

```

OpenSSLで表示されるCisco IOS XEデバイスの証明書情報

Cisco IOS XEデバイスの証明書情報を確認します。

コマンド `show crypto pki certificates verbose` を使用すると、デバイスで使用可能なすべての証明書の証明書情報が出力されます。

```

9800#show crypto pki certificates verbose
CA Certificate <-----Type of certificate
  Status: Available
  Version: 3
  Certificate Serial Number (hex): 2A352E27C69021ECE1AA61751CA1F233E0636FB1
  Certificate Usage: General Purpose
  Issuer: <-----DN for issuer
    cn=RootCA
    ou=Cisco Wireless
    o=Cisco lab
    l=CDMX
    st=CDMX

```

```
c=MX
Subject: <-----DN for subject
  cn=RootCA
  ou=Cisco Wireless
  o=Cisco lab
  l=CDMX
  st=CDMX
  c=MX
Validity Date: <-----Validity date
  start date: 14:54:02 Central Jul 22 2024
  end date: 14:54:02 Central Jul 20 2034
Subject Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (2048 bit) <-----Key size
Signature Algorithm: SHA256 with RSA Encryption
Fingerprint MD5: 432021B5 B4BE15F5 A537385C 4FAB9A94
Fingerprint SHA1: 86D18427 BE619A2A 6C20C314 9EDAAEB2 6B4DFE87
X509v3 extensions:
  X509v3 Subject Key ID: 57DEEBD8 3214CA05 176FOCD6 6C842EBC 9ABFF7D8
  X509v3 Basic Constraints:
    CA: TRUE
  X509v3 Subject Alternative Name:
    RootCA <-----SANS
    IP Address :
    OtherNames :
  X509v3 Authority Key ID: 57DEEBD8 3214CA05 176FOCD6 6C842EBC 9ABFF7D8
  Authority Info Access:
Cert install time: 16:42:09 Central Jul 22 2024
Associated Trustpoints: WLC.pfx-rrr1 <-----Associated trustpoint
Storage: nvram:RootCA#6FB1CA.cer
```

トラブルシューティング

失効チェックが実行されている

証明書がCisco IOS XEにインポートされると、新しく作成されたトラストポイントの失効チェックが有効になります。インポートされた証明書トラストポイントを検証に使用する必要があるデバイスに証明書が提示されると、デバイスは存在しない証明書失効リスト(CRL)を検索し、失敗します。メッセージが端末に表示されます。

```
Jul 17 21:50:39.068: %PKI-3-CRL_FETCH_FAIL: CRL fetch for trustpoint WLC1.pfx failed
Reason : Enrollment URL not configured.
```

証明書の検証パスにある各トラストポイントに、`revocation-check none` コマンドが含まれていることを確認します。

関連情報

- [Catalyst 9800 WLCでのCSR証明書の生成とダウンロード](#)
- [IOS XE PKIによるCA署名付き証明書の設定](#)
- [セキュリティおよびVPNコンフィギュレーションガイド、Cisco IOS XE 17.x](#)
- [9800 WLCのチェーンを作成するための証明書情報について](#)

翻訳について

シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。