Generate Certificate Using OpenSSL Only

Certificate Generation Using OpenSSL Only

This section describes the process for generating a private key and certificate request for the Expressway using OpenSSL. This is a generic process that relies only on the free OpenSSL package and not on any other software. It is appropriate when certificates are required to interface with neighboring devices for test purposes, and provide output to interact with Certificate Authorities.

The output for the certificate request generation process is given to a Certificate Authority which may be internal or external to the organization, and which is used to produce the X.509 certificates required by the Expressway to authenticate itself with neighboring devices.

This section also briefly describes how OpenSSL is used to manage a private Certificate Authority, but does not intend to be comprehensive. Various components of these processes are used when interfacing with third party CAs.

OpenSSL and Mac OS X or Linux

OpenSSL is already installed on Mac OS X, and is usually installed on Linux.

OpenSSL and Windows

If you do not have OpenSSL already installed, this is available as a free download from http://www.openssl.org/related/binaries.html.

Choose the relevant 32 bit or 64 bit OpenSSL - the ‘Light’ version is all that is needed.

If you receive a warning while installing OpenSSL that C++ files cannot be found, load the “Visual C++ Redistributables” also available on this site and then re-load the OpenSSL software.

Create a Certificate Request Using OpenSSL

This process creates a private key and certificate request for the server that is validated by a CA. This could be a CA that is created and managed locally, or a third-party CA.

Note

- This method to create a CSR should only be used if you have a good knowledge of working with OpenSSL as there is a potential for entering incorrect commands (especially with numerous SAN entries). Missing relevant SAN entries would require recreating the certificate at a later date.
- From version X8.5.1, the user interface provides an option to set the Digest algorithm. The default is set to SHA-256, with options to change to SHA-1, SHA-384, or SHA-512.

To generate the CSR from the command line with OpenSSL use these instructions:

Procedure

Step 1 SSH to the Expressway and log in as root.

Step 2 Make a new directory to do the work in - `mkdir /tmp/certtemp`

Step 3 Move in to this directory - `cd /tmp/certtemp`

Step 4 Copy the Open SSL configuration file we use for CSR to this directory, as we need to edit it **(Note: Keep the dot at the end)** - `cp /etc/openssl/csrreq.cnf`

Step 5 Open the file for editing – `vi csrreq.cnf`

Step 6 Find the line "`default_md = sha1`" and edit it so that it reads "`default_md = sha256`"

Step 7 Uncomment the line "`# req_extensions = v3_req`" by removing the # at the start of it

Step 8 Make sure that the line "`extendedKeyUsage=serverAuth, clientAuth`" is present within the section `[v3_req]`

Step 9 Find the line "`subjectAltName = ${ENV::CSR_ALT_NAME}`" and replace it such that it lists what you want in the Subject Alternative Names in the certificate e.g. "`subjectAltName = DNS:peer1vcs.example.com,DNS:peer2vcs.example.com,DNS:ClusterFQDN.example.com`". Make sure you add all the additional relevant entries. For MRA this may comprise:

a. Expressway E: `DNS:<CM domain name>, DNS:<XMPP federation domain>, DNS:<federation chat alias 1>, DNS:<federation chat alias 2>`, etc.

b. Expressway C: `DNS:<secure profile name 1>, DNS:<secure profile name 2>`, etc.

Step 10 Now save the file and exit.

Step 11 Run the following OpenSSL command to generate a new CSR and Private key for the VCS "`openssl req -nodes -newkey rsa:4096 -keyout privatekey.pem -out myrequest.csr -config csrreq.cnf`" changing the `rsa:nnnn` if required. (nnnn = keylength, recommended number is 4096).

Step 12 The console displays output similar to the following example, where you are required to enter information. You do not need to populate all of them, but some fields are required:

- Country
- State and Province

- Locality name
- Organization name
- Common name
- Email address - optional, can leave blank
- A challenge password - optional, can leave blank
- An optional company name - optional, can leave blank

```
Generating a 4096 bit RSA private key
.............++
.................................................++
writing new private key to 'privatekey.pem'
-----
You are about to be asked to enter information that will be incorporated into your certificate
request.
What you are about to enter 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 (2 letter code) [AU]:GB
State or Province Name (full name) [Some-State]:Berkshire
Locality Name (eg, city) []:Reading
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cisco
Organizational Unit Name (eg, section) []:CIBU
Common Name (eg, YOUR name) []:exp01.example.com
Email Address []:
```

When you have completed the fields, you will have two new files, **myrequest.csr** and **privatekey.pem**.

Step 13 (Optional) If you want to validate the DNS entries have been entered correctly into the request, the **myrequest.csr** file can be decoded using the command: `openssl req -text -noout -in myrequest.csr`

Step 14 Submit the CSR to your chosen Certificate Authority, who will provide the public certificate.

Step 15 Upload the public certificate to the VCS via **Maintenance** > **Security** > **Server certificate** webpage, "**Select the server certificate file**" entry box.

Step 16 Upload the **privatekey.pem** to the VCS via **Maintenance** > **Security** > **Server certificate** webpage, "**Select the server private key file**" entry box.

The **privatekey.pem** should be kept safe.

Operate as a Certificate Authority Using OpenSSL

A major deployment is to make use of a third-party certificate authority, or already have one internal to an organization's IT department. However, you can use OpenSSL to manage certificates in a private certificate authority as outlined below.

If you have already configured OpenSSL to act as a CA, go to section Create Signed Certificate Using OpenSSL.

Configure OpenSSL Act as CA

OpenSSL is powerful software, and when operating as a CA, requires a number of directories and databases to be configured for tracking issued certificates.

The list of directories and files can be found in the openssl configuration file under the section `[ CA_default ]`. By default, create the required files/directories:

- A **demoCA** directory in the current directory, with 3 subdirectories **certs**, **newcerts**, and **private**.
- An empty file called **index.txt** in the **demoCA** directory.
- A file called **serial** in the **demoCA** directory, storing a 2-digit number, such as "10".

For example, use the commands:

```
mkdir demoCA

cd demoCA

mkdir certs

mkdir newcerts

mkdir private

touch index.txt

echo 10 > serial
```

Create Certificate Authority Using OpenSSL

This process creates a private key and certificate of a Certificate Authority (CA), which is used to validate other certificates. Note that this will not be trusted by devices outside of those on which it is explicitly installed.

From a command prompt:

Procedure

Step 1 Ensure that you are in the **demoCA** directory.

Step 2 For Windows: copy **openssl.cfg** from the directory where OpenSSL is installed to the **demoCA** directory and rename it as **openssl_local.cfg**.

For Mac OS X: copy **/System/Library/OpenSSL/openssl.cnf** to the **demoCA** directory and rename it as **openssl_local.cfg**.

Step 3 Use a text editor to edit the **openssl_local.cfg** file that was created by the above copy command. Make the following modifications to the `[CA_default]` section:

a. Ensure that the line `copy_extensions = copy` does not have a # at the beginning of the line. Delete the # if it is there. If the line remains commented out, it will strip attributes in the CSR and, SSL Server and SSL Client attributes will not appear in the certificate.

b. Change `policy = policy_match` to `policy = policy_anything`

c. Change `dir = ./demoCA` to `dir = .`

d. Optionally, change `default_days = 365` (1 year validity of the generated certificate) to `default_days = 3650` (10 years, or choose another suitable value).

e. Save the file.

Step 4 Generate a private key for the CA by running the following command:

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

This prompts for a password to encrypt the private key: choose a strong password and record it in a safe place. The cakey.pem file is used to create the CA certificate and to sign other certificates and must also be kept secure.

Step 5 Generate the CA certificate by running the following command.

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

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

Step 6 Enter a passphrase for the key, and then enter the data requested, including:

- Country
- State or Province
- Locality name
- Organization name
- Organizational unit
- Common name - this is typicaly the name of the contact person for this CA
- Email address - optional, can leave blank

After you enter the requested data, the operation is complete and the certificate authority certificate **cacert.pem** is now available.

Create Signed Certificate Using OpenSSL

This process signs the server certificate with the generated CA key, using previously generated certificate request.

From a command prompt:

Procedure

Step 1 Ensure that you are in the **demoCA** directory.

Step 2 Ensure that the certificate request file **(certcsr.pem)** is available:

- If the certificate request is created using the Expressway (recommended process):

  Copy the file downloaded from the Expressway into the **demoCA** directory and rename it as **certcsr.pem**.

- If the certificate request is created using OpenSSL:

  Copy the previously generated certificate request into the **demoCA** directory and then covert it to PEM format by running the following command:

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

Step 3 Generate a signed server certificate by running the following command:

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

If you receive a "failed to update database TXT_DB error number 2" error message, you can remove the contents of the index.txt file and then rerun the command.

Step 4 You will be prompted to enter the password for the CA's private key.

The signed certificate for the server is now available as **demoCA/certs/server.pem**.

Create Self-Signed Certificates Using OpenSSL

We do not recommend creating self-signed certificates. They will not work in Unified Communications deployments.

Instead, you should create a Certificate Authority using OpenSSL as described above.