Managing security certificates with OpenSSL

SMP Gateway devices use a default X.509 certificate to enable secure SSL/TLS communication for web access, device maintenance and protocol communications. The default X.509 certificate is required during the SSL/TLS handshake to initiate a secure encrypted communication; it however does not ensure strong device authentication.

It is good practice to replace the provided default certificate with a custom one created specifically for each device. Typically, certificates are generated from a corporate Public Key Infrastructure (PKI), but such an infrastructure may not be available to every organization.

The purpose of this document is to provide technical instructions on how to generate and maintain device certificates using the free open source OpenSSL utility in the Microsoft Windows environment.

Quebec City
1990 5th Street
Suite 220
Levis, Quebec
Canada G6W 5M6
Phone: +1.418.830.5800
Toll free: +1.877.834-0009
Fax: +1.514.227.5256

Montreal
1155 Boulevard Robert-Bourassa
Suite 1215
Montreal, Quebec
Canada H3B 3A7
Phone: +1.514.845.6195
Toll free: +1.877.834-0009
Fax: +1.514.227.5256

www.Eaton.com/smartgrid

Contents

Powering Business Worldwide

Setting up your environment

The open source OpenSSL package is available as source code at the OpenSSL web site https://www.openssl.org/. OpenSSL in binary form must be obtained from third party distributions. A list of third party distributions is available at the following link: https://wiki.openssl.org/index.php/Binaries.

Eaton recommends installing the latest "Light" version of Win32 OpenSSL binaries, which is available at the following link: http://slproweb.com/products/Win32OpenSSL.html.

This distribution is a simple and light 3MB installation of the OpenSSL binaries for the Windows environment. The installation is straight-forward; simply follow the installation procedure using the default options. This present document assumes that Win32 OpenSSL v1.1.0f_Light or later version of the Win32 OpenSSL binaries was installed. A 64-bit version is also available, but it is assumed that the 32-bit version is installed in the rest of the document.

Once the installation is completed, go to the "OpenSSL-Win32/bin" folder. This folder should contain the **openssl.exe** utility, along with the **openssl.cfg** configuration file.

- Rename, move or preferably delete the default **openssl.cfg** configuration file of the installer. This file should never be referred to when issuing OpenSSL commands, and will be replaced by a new customized one.

- The Appendix of this document provides the content of the **openssl.cfg** file in plain text. We also provide a link to the **openssl.txt** file provided by Eaton; once you open the file, rename it **openssl.cfg**.

- Copy the renamed **openssl.cfg** configuration file supplied with the link to the "OpenSSL-Win32/bin" folder. The new configuration file is customized to match the OpenSSL commands of this document with appropriate settings. The content of the **openssl.cfg** file is also available in the Appendix section of this document.

- Using a text editor, like Notepad, modify the following section of the **openssl.cfg** file with the information related to your organization. These values will be the defaults values for the certificate Distinguished Name when prompted.

```
# Default values for the above. To be replaced by your information
countryName_default                = US
stateOrProvinceName_default        = My State
0.organizationName_default         = My Company
organizationalUnitName_default     = My Division
localityName_default               = My City
```

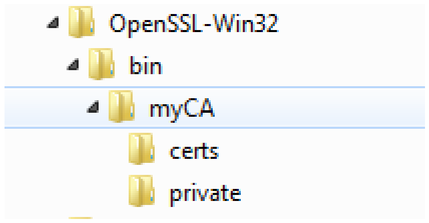
- Specify the **openssl.cfg** configuration file location in Windows environment variable, otherwise the default C:\Program Files (x86)\Common\Files\SSL\openssl.cnf file will be searched for. It is very important that the **openssl.cfg** file supplied with this document is found by the openssl utility.
    - Add `OPENSSL_CONF=C:\OpenSSL-Win32\bin\openssl.cfg` in global system environment variables

  OR

    - Open a Command Prompt (CMD) as Administrator
    - Run the following command:
      `SET OPENSSL_CONF=C:\OpenSSL-Win32\bin\openssl.cfg`

Note: Subsequent openssl commands must be performed in this Command Prompt session

- Create a directory tree, as shown below. These folder names are already configured in the **openssl.cfg** file provided with this document. The folder names can be personalized but they must also be updated in the **openssl.cfg** file, otherwise the procedure won't work.



- Create an empty file named **index.txt** in the **myCA** folder.
- Create an empty file named **index.txt.attr** in the **myCA** folder.
- Create a text file for the serial number and name it **serial.srl**; the file should be placed in the **myCA** folder. The content of the file is a single line with the serial number of the first created certificate (for example: `01`). The Serial number will be automatically incremented after each certificate creation. Note that the serial number must be in hexadecimal format if a value other than "01" is written in the text file.

Managing the validity period of certificates

The validity period of a certificate corresponds to the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field presents two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter). Both **notBefore** and **notAfter** dates may be encoded as UTC time or Generalized time (ASN1 Time definition), depending on the dates.

By default, when you create a certificate with this procedure, the **notBefore** date is the date you are creating the certificate and the **notAfter** date is the **notBefore** date plus five (5) years (1825 days). This is configured in the **default_days** setting of the openssl.cfg file.

You might want to change the duration of the validity period. To do so, modify the **default_days** setting in the openssl.cfg file. In this case, **notBefore** date will still be the date you are creating the certificate and **notAfter** date will be the **notBefore** date plus the newly configured number of days.

Alternatively, you might want to change the **notBefore** and **notAfter** dates specificaly. It is recommended to set a **notBefore** prior to the date of creation of the certificate since this date can lead to a certificate refusal. The certificate is considered not yet valid when used immediately after creation. To modify both dates, add the **-startdate** and **-enddate** parameters when signing the certificate with the `openssl ca` command described in the following sections. Both **-startdate** and **-enddate** parameters must be used together.

If the date is set for 2050 or earlier, it can be specified in ASN1 UTCTime format: `YYMMDDHHMMSSZ`. For example: `200101120000Z` is for January 1st 2020.

If the date is set for 2050 or later, it must be specified in ASN1 GeneralizedTime format: `YYYYMMDDHHMMSSZ`. For example `20600101120000Z` is for January 1st 2060.

More examples:

```
openssl ca -startdate 200101120000Z -enddate 250101120000Z -in
myDevice.csr -out myDevice.crt

openssl ca -startdate 20550101120000Z -enddate 20650101120000Z -in
myDevice.csr -out myDevice.crt
```

Creating a Self-Signed Certificate (SSC)

A Self-Signed Certificate (SSC) is a public key certificate signed by its own private key. The purpose of a SSC should be solely to enable a SSL/TLS handshake where the server must provide a public key certificate to the client. SSC does not ensure strong device authentication; it however enables SSL/TLS handshake with session key exchange for data encryption.

A Self-Signed Certificate (SSC) should not be used to sign device certificates, this is the role of a Public Key Infrastructure based on a root Certificate Authority (CA) and subordinate CAs. Therefore, the SSC created with this procedure will not have the basic constraint of type CA in the certificate extensions.

The device certificate must be configured into the SMP Gateway in pkcs12 format. The last step below explains how to generate this pkcs12 file from the certificate and key files.

For each command, ensure you never get the following error message; if so, verify and set the OPENSSL_CONF environment variable as described above in the **Setting up your environment** section.

```
WARNING: can't open config file: <path>/openssl.cnf
```

Open a command line session with “OpenSSL-Win32/bin” as the current folder. Enter the OpenSSL commands described below.

- Generate a 2048-bit length RSA key (myDeviceSSC.key) that is password protected.

```
openssl genrsa -aes256 -out myDeviceSSC.key 2048

C:\OpenSSL-Win32\bin>openssl genrsa -aes256 -out myDeviceSSC.key 2048
Generating RSA private key, 2048 bit long modulus
....+++
...................................................................+++
e is 65537 (0x010001)
Enter pass phrase for myDeviceSSC.key:
Verifying - Enter pass phrase for myDeviceSSC.key:
```

- Generate and auto-sign the SSC using its own private key. The certificate is valid for 5 years (1826 days) in the example below. It is important to specify the “v3_ssc“ extensions section of the openssl.cfg file. When prompted, enter your organization values in the Distinguished Name fields. The default values are the ones configured in the openssl.cfg file. Specify a Common Name value that is a unique device name or IP address for the device.

```
openssl req -x509 -extensions v3_ssc -new -key myDeviceSSC.key -sha256 -
days 1826 -out myDeviceSSC.crt

C:\OpenSSL-Win32\bin>openssl req -x509 -extensions v3_ssc -new -key
myDeviceSSC.key -sha256 -days 1826 -out myDeviceSSC.crt
Enter pass phrase for myDeviceSSC.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 (2 letter code) [US]:US
State or Province Name (full name) [My State]:My State
Organization Name (company) [My Company]:My Company
Organizational Unit Name (department, division) [My Division]:My Division
Common Name (server FQDN, device name or IP address) []:myDeviceSSC
```

- Create a pkcs12 file format that will contain the SSC public certificate along with its private key.

```
openssl pkcs12 -export -out myDeviceSSC.p12 -inkey myDeviceSSC.key
-in myDeviceSSC.crt
```

Creating a root Certificate Authority (CA)

The first step in creating a chain of trust is to create a root Certificate Authority (CA). The root CA is used to digitally sign subordinate certificate authorities or end certificates.

For each command, ensure you never get the following error message; if so, verify and set the OPENSSL_CONF environment variable as described above in the **Setting up your environment** section.

```
WARNING: can't open config file: <path>/openssl.cnf
```

- Generate a 2048-bit long RSA key that is password protected. This key must be properly protected against divulgation, otherwise every certificate issued by this CA key will need to be revoked.

```
openssl genrsa -aes256 -out rootCA.key 2048

C:\OpenSSL-Win32\bin>openssl genrsa -aes256 -out rootCA.key 2048
Generating RSA private key, 2048 bit long modulus
....................................+++
.................+++
e is 65537 (0x010001)
Enter pass phrase for rootCA.key:
Verifying - Enter pass phrase for rootCA.key:
```

- Generate and auto-sign the root CA public certificate using its own private key. The rootCA certificate is valid for 5 years (1826 days) in the example below. Enter you own organization related values in the Distinguished Name. Specify a unique Common Name, here “myRootCA”.

```
openssl req -x509 -new -key rootCA.key -sha256 -days 1826 -out rootCA.crt

C:\OpenSSL-Win32\bin>openssl req -x509 -new -key rootCA.key -sha256
-days 1826 -out rootCA.crt
Enter pass phrase for 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 (2 letter code) [US]:US
State or Province Name (full name) [State or Province]:My State
Organization Name (company) [My Company]:My Company
Organizational Unit Name (department, division) [My Division]:My Division
Common Name (server FQDN, device name or IP address) []:myRootCA
```

- Verify the certificate content with this next command. This ensures that the custom **openssl.cfg** file was properly found and used when creating the certificate extensions.
    - o Certificate version is 3
    - o Signature algorithm is **sha256WithRSAEncryption**
    - o Basic Constraint extension is **CA:TRUE, pathlen:1**
    - o Key Usage is **Certificate Signing** only

```
openssl x509 -in rootCA.crt -noout –text
```

- Move the root CA public certificate **rootCA.crt** to **myCA** folder.
- Move the root CA private key **rootCA.key** to the **myCA\private** folder.

Creating a subordinate Certificate Authority

It is not mandatory but good practice to create a subordinate CA to sign device certificates. To create a subordinate CA, a Certificate Signing Request (CSR) must be issued, and then processed to sign the certificate by the root CA.

If you do not intend to create a subordinate CA, skip this section and go to the **Creation of Device Certificates** section.

- Generate a 2048-bit long RSA subCA key that is password protected.

```
openssl genrsa -aes256 -out subCA.key 2048
```

- Create a Certificate Signing Request (CSR) for the subCA. Specify a unique Common Name, here “mySubCA”.

```
openssl req -new -key subCA.key -sha256 -out subCA.csr

C:\OpenSSL-Win32\bin>openssl req -new -key subCA.key -sha256 -out subCA.csr
Enter pass phrase for subCA.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 (2 letter code) [US]:US
State or Province Name (full name) [State or Province]:My State
Organization Name (company) [My Company]:My Company
Organizational Unit Name (department, division) [My Division]:My Division
Common Name (server FQDN, device name or IP address) []:mySubCA
```

- Optionally, visualize the CSR content.

```
openssl req -text -noout -verify -in subCA.csr
```

- Process the CSR by creating and signing the subordinate CA public certificate with the rootCA key. The subCA certificate is valid for 5 years (1826 days) in the example below.

```
openssl ca -in subCA.csr -extensions v3_subca -out subCA.crt -days 1826

C:\OpenSSL-Win32\bin>openssl ca -in subCA.csr -extensions v3_subca -out
subCA.crt -days 1826
Using configuration from C:\OpenSSL-Win32\bin\openssl.cfg
Enter pass phrase for ./myCA/private/rootCA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'My State'
organizationName      :PRINTABLE:'My Company'
organizationalUnitName:PRINTABLE:'My Division'
commonName            :PRINTABLE:'mySubCA'
```

```
Certificate is to be certified until Sep  4 17:52:07 2022 GMT (1826 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entry
Data Base Updated
```

- Move the subordinate CA public certificate **subCA.crt** to **myCA** folder.
- Move the subordinate CA private key **subCA.key** to the **myCA\private** folder.
- Delete the intermediate **subCA.csr** CSR file.

Creating device certificates

Device certificates can be created and signed with the root CA or the subordinate CA. The device certificate duration is 5 years by default, as configured in **openssl.cfg** file. The expiration date can be modified by setting the **default_days** attribute in **openssl.cfg** file; however, it shall not exceed the expiration date of one of its CA.

OpenSSL creates a database of certificates issued by the CA. By default, a certificate with the same Distinguished Name cannot be issued a second time without being revoked. To reissue the same certificate, the "**unique_subject**" attribute in the **myCA/index.txt.attr** file must be set to "**no**".

The device certificate must be configured into the SMP Gateway in pkcs12 format. The following steps explain how to generate a pkcs12 file from the certificate and key files.

- Generate a 2048-bit long RSA key that is password protected.

```
openssl genrsa -aes256 -out myDevice.key 2048

c:\Program Files\OpenSSL-Win64\bin>openssl genrsa -aes256 -out
myDevice.key 2048
Generating RSA private key, 2048-bit long modulus
...................+++
........................+++
e is 65537 (0x010001)
Enter pass phrase for myDevice.key:
Verifying - Enter pass phrase for myDevice.key:
```

- Create a CSR for the device certificate. Specify a unique Common Name, here "myDevice".

```
openssl req -new -key myDevice.key -sha256 -out myDevice.csr

C:\Program Files\OpenSSL-Win64\bin>openssl req -new -key myDevice.key -
sha256 -out myDevice.csr
Enter pass phrase for myDevice.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 (2 letter code) [US]:US
State or Province Name (full name) [State or Province]:My State
Organization Name (company) [My Company]:My Company
Organizational Unit Name (department, division) [My Division]:My Division
Common Name (server FQDN, device name or IP address) []:myDevice
```

- Process the CSR by creating and signing the device certificate with the subCA or rootCA key.
    - To sign with the default rootCA (the root CA files are defined in openssl.cfg file):

```
openssl ca -in myDevice.csr -out myDevice.crt

C:\OpenSSL-Win32\bin>openssl ca -in myDevice.csr -out myDevice.crt
Using configuration from C:\OpenSSL-Win32\bin\openssl.cfg
Enter pass phrase for ./myCA/private/rootCA.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'My State'
organizationName      :PRINTABLE:'My Company'
organizationalUnitName:PRINTABLE:'My Division'
commonName            :PRINTABLE:'myDevice'
Certificate is to be certified until Sep  5 13:27:06 2022 GMT (1826
days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entry
Data Base Updated
```

    - To sign with the subordinate CA:

```
openssl ca -in myDevice.csr -cert myCA/subCA.crt -keyfile
myCA/private/subCA.key -out myDevice.crt
```

- Create a pkcs12 file containing the private key, the public certificate and the subordinate CA if applicable.
    - To create the pkcs12 file for a device certificate signed by a root CA:

```
openssl pkcs12 -export -out myDevice.p12 -inkey myDevice.key -in
myDevice.crt
```

    - To create the pkcs12 file for a device certificate signed by the subordinate CA, the subCA certificate must be specified for inclusion into the file:

```
openssl pkcs12 -export -out myDevice.p12 -inkey myDevice.key
-in myDevice.crt –certfile myCA/subCA.crt
```

- To verify the content of the pkcs12 file.

```
openssl pkcs12 -info -in myDevice.p12
```

Creating OPC UA certificates

The SMP Gateway implements OPC UA master and slave protocols (client and server). If a security policy is used, an Application Instance Certificate (X.509 v3 certificate) is mandatory to identify the application's URI (Universal Resource Identifier) when connecting to other OPC UA Applications. Each Application Instance has a globally unique URI that must be added to the certificate in the **Subject Alternative Name** field.

A Self-Signed Certificate (SSC) or a Device Certificate can be used as an OPC UA Application Instance Certificate. To do so, modify the **openssl.cfg** file as follow:

- For a Self-Signed Certificate (SSC):
    - uncomment the **#subjectAltName** parameter in the **[v3_ssc]** section
    - uncomment the **#URI** parameter in the **[alt_names]** section and set <YourResourceIdentifier> to your specific Application Instance ID

- For a Device Certificate:
    - uncomment the **#subjectAltName** parameter in the **[v3_req]** section
    - uncomment the **#URI** parameter in the **[alt_names]** section and set <YourResourceIdentifier> to your specific Application Instance ID

Note: An OPC UA Application installed on a single machine is called an Application Instance. Each instance has its own Application Instance Certificate which it uses to identify itself when connecting to other OPC UA Applications (the Public Key and Private Key).

Once the **openssl.cfg** file is modified, follow the procedure in the following sections to create your OPC UA certificate:

- **Creating a Self-Signed Certificate (SSC)** for a self-signed
- **Creating device certificates** for a device certificate

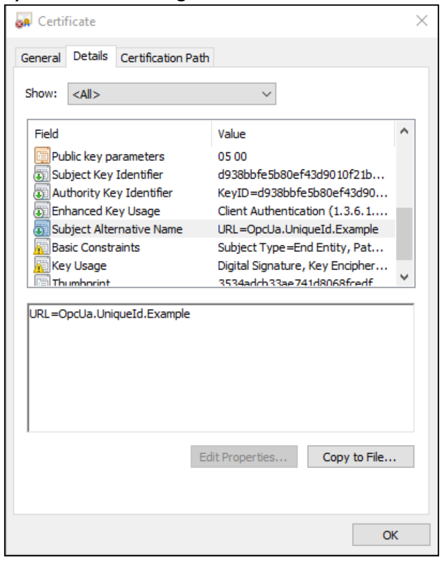
Example for a Self-Signed Certificate for OPC UA:

1) Modification to the file (partially shown):

```
[ v3_ssc ]
basicConstraints        = critical, CA:false
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid, issuer
keyUsage                = critical, keyEncipherment, digitalSignature # dataEncipherment, keyAgreement,
                          encipherOnly, decipherOnly
extendedKeyUsage        = clientAuth, serverAuth                      # codeSigning, emailProtection,
                          timeStamping, OCSPSigning, ipsecIKE
subjectAltName         = @alt_names
#nsComment              = "OpenSSL Generated Certificate"
----------------------------------------------------------------------------------------------------

[ alt_names ]           # Uncomment below to add IP address, DNS and/or URI. At least one must be
                          declared if @alt_names
                        # is referred
#IP.1                    = 123.123.123.123
#IP.2                    = 123.123.123.123
#DNS.1                   = <YourDomainName>
#DNS.2                   = <YourDomainName2>
URI                     = OpcUa.UniqueId.Example
```

2) The resulting certificate:



Field	Value
Public key parameters	05 00
Subject Key Identifier	d938bbfe5b80ef43d9010f21b...
Authority Key Identifier	KeyID=d938bbfe5b80ef43d90...
Enhanced Key Usage	Client Authentication (1.3.6.1....
Subject Alternative Name	URL=OpcUa.UniqueId.Example
Basic Constraints	Subject Type=End Entity, Pat...
Key Usage	Digital Signature, Key Encipher...
Thumbprint	3534adcb33ae741d8068fcedf

URL=OpcUa.UniqueId.Example

Note that in the previous image, the certificate show URL and not URI for the Subject Alternative Name, this is caused by Microsoft viewer, if you open your certificate with the viewer of OpenSSL, you will see URI.

Do not forget to revert your OPC UA-specific changes in the **openssl.cfg** file to allow you to create other types of certificates.

Getting assistance

If you have any questions regarding the performance, application, testing or repair of this or any other component of the SMP product line, do not hesitate to contact us. Our staff will be happy to assist you.

Technical Support
Eaton's Energy Automation Solutions

Email:	eas-support@eaton.com
Phone:	+1.877.834-0009 or +1.800.815-2258

Business hours are from 8 a.m. and 5 p.m. CST, Monday to Friday.

Appendix – openssl.cfg content

```
#
# OpenSSL configuration file.
#
#

[ ca ]
default_ca          = CA_default

[ CA_default ]
dir                 = ./myCA            # CA root directory
certificate         = $dir/myRootCA.crt # CA public key certificate, same as -cert command line
private_key         = $dir/private/myRootCA.key   # CA private key, same as -keyfile command line
serial              = $dir/serial.srl   # Text file containing the next serial number to use in hex
                                        # Created automatically with -create_serial command line
database            = $dir/index.txt    # Text database file of issued certificates
                                        # This file must be present though initially it will be empty
new_certs_dir       = $dir/certs        # Where issued certificates are kept, same as -outdir command line
certs               = $new_certs_dir    # Depending on openssl version, this setting might be used instead of new_certs_dir

#RANDFILE           = $dir/private/random.rnd   # File used to read and write random number seed information
x509_extensions     = v3_req                    # The extensions added when the CA signs a request

# CRL support; settings defined but not used
#crl_dir            = $dir/crls                 # Where issued crl are kept
#crlnumber          = $dir/crlnumber.crl        # Text file containing the next CRL number to use in hex
                                                # The crl number will be inserted in the CRLs only if this file exists
#crl                = $dir/crl.crt              # The current CRL

default_days        = 9131              # Number of days to certify the cert for (5 years)
default_crl_days    = 30                # How long before next CRL
default_md          = sha256            # Use SHA256 for digital signatures (MD - Message Digest)
preserve            = no                # Don't re-order the DN
email_in_dn         = no                # Don't add the EMAIL field to the DN
nameopt             = default_ca
certopt             = default_ca
policy              = policy_match

[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits            = 2048          # Size of keys
#default_keyfile        = newkey.pem    # name of generated keys
#default_md             = sha256        # Use SHA256 for digital signatures (MD - Message Digest)
string_mask             = nombstr       # permitted characters, non-UTF printable strings
distinguished_name      = req_distinguished_name
x509_extensions         = v3_ca         # The extensions to add to self signed certificates
req_extensions          = v3_req        # The extensions to add to end certificates (user or device)

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_min         = 2
countryName_max         = 2
stateOrProvinceName     = State or Province Name (full name)
0.organizationName      = Organization Name (company)
organizationalUnitName  = Organizational Unit Name (department, division)
#localityName           = Locality Name (city, district)
commonName              = Common Name (server FQDN, device name or IP address)
commonName_max          = 64

# Default values for the above. To be replaced with the information of your organization
countryName_default             = US
stateOrProvinceName_default     = My State
0.organizationName_default      = My Company
organizationalUnitName_default  = My Division
localityName_default            = My City

[ v3_ca ]
basicConstraints        = critical, CA:true, pathlen:1              # Can sign up to 1 subordinate CA
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always, issuer:always
keyUsage                = critical, keyCertSign, digitalSignature    # nonRepudiation, cRLSign
#nsComment              = "OpenSSL Generated Certificate"

[ v3_subca ]
basicConstraints        = critical, CA:true, pathlen:0              # Cannot sign subordinate CA, only end certificates
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always, issuer:always
keyUsage                = critical, keyCertSign, digitalSignature    # nonRepudiation, cRLSign
#nsComment              = "OpenSSL Generated Certificate"
```

```
[ v3_ssc ]
basicConstraints        = critical, CA:false
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid, issuer
keyUsage                = critical, keyEncipherment, digitalSignature # dataEncipherment, keyAgreement, encipherOnly,
                                                              # decipherOnly
extendedKeyUsage        = clientAuth, serverAuth                      # codeSigning, emailProtection, timeStamping, OCSPSigning,
ipsecIKE
#subjectAltName         = @alt_names
#nsComment              = "OpenSSL Generated Certificate"

[ v3_req ]
basicConstraints        = critical, CA:false
subjectKeyIdentifier    = hash
keyUsage                = critical, keyEncipherment, digitalSignature # dataEncipherment, keyAgreement, encipherOnly,
                                                              # decipherOnly
extendedKeyUsage        = clientAuth, serverAuth                      # codeSigning, emailProtection, timeStamping, OCSPSigning,
                                                              # ipsecIKE
#subjectAltName         = @alt_names
#nsComment              = "OpenSSL Generated Certificate"

[ alt_names ]           # Uncomment below to add IP address, DNS and/or URI. At least one must be declared if @alt_names
                        # is referred
#IP.1                    = 123.123.123.123
#IP.2                    = 123.123.123.123
#DNS.1                   = <YourDomainName>
#DNS.2                   = <YourDomainName2>
#URI                     = <YourResourceIdentifier>
```

The **Openssl.txt** file is available in the same folder as the present Technical Note.