HP E-Series MSM Controllers - How to create SSL Certificates

Technical white paper Version 3.03 – March 14, 2011

Click here to download a zip file with this paper, sample certs and SSL scripts

Table of contents

Overview of SSL certificates 2
- Outline of SSL Web certificate creation examples 3
- SSL Authentication and Trust Sequence 4
- Installing OpenSSL for Windows 5
- Creating a Certificate Signing Request (CSR) 6
- Creating your own CA Certificate 9
- What does a certificate look like ? 12
- Creating an SSL certificate with your CA certificate 13
- Creating a certificate chain 15
- Converting a .PEM file to PKCS #12 format 17
- Installing the new SSL certificate onto the MSM 18
- Where can I buy an SSL certificate ? 18
- Certificate Revocation List (CRL) 19

Overview of SSL certificates

SSL certificates that are installed on a web server are intended primarily to provide legitimacy and validation that the web server that you visit can be a trusted source. They also provide a secure, private SSL (Secure Sockets Layer) encrypted connections to the web server for banking, logins and other sensitive transactions.

The E-Series Mobility MSM products run a web server on the controllers to provide an HTML-based Login Portal for users with web browsers as a way to authenticate, and as a way to manage the products. The Mobility MSM products come with a default "sample" certificate installed. Although this certificate is not "trusted" by the browser, it still permits you the opportunity to "accept" the resulting certificate warning and proceed to make a secure SSL encrypted connection to the controller, either as the administrator of the unit, or as a HTML guest user.

Most customers feel it's not desirable to see these confusing and unsightly certificate warning messages pop up when a guest wants to login through the Mobility MSM. This certificate warning appears because a user's browser's CA (Certificate Authority) certificate store does not contain a CA certificate that can validate the certificate on the Mobility MSM, thus making it trusted by the browser. (The browser CA certificate store actually would have a copy of the CA installed).

The solution is to install a trusted valid SSL certificate on the Mobility MSM controller that already has the matching CA certificate installed in the highest percentage of browsers used today.

To avoid presenting guest users with a certificate warning every time they login, it's expected that you'll install a valid SSL certificate on the Mobility controller to permit secure SSL connections to client browsers to be established, without triggering any certificate warnings.

You can purchase a valid SSL certificate from Certification Authorities, such as Entrust, Verisign, Thawte and other certificate vendors, such as GoDaddy. All you need to do is send them a CSR (Certificate Signing Request). This is explained later.

Outline of SSL Web certificate creation examples

The following three examples for creating SSL certificates are described in this document:

- **Create a Certificate Signing Request (CSR) for an SSL certificate:** This is what you send to the certificate vendor to get your certificate signed. This is the best option, since it ensures that the highest percentage of web browsers can validate your SSL certificate. A number of certificate vendors offer this service for a nominal charge. These include certificate authorities like Thawte, Verisign, and Entrust.

- **Create your own Private CA certificate:** This is a self-signed CA certificate that you create. It is intended to be used to sign CSRs of your own, but since the CA certificate is home-made, your CSR, once signed by this CA, will only be trusted by this CA. Self-signed CA certificates and the SSL certificates that they sign should generally be used for testing purposes only. They may also represent your Private Key Infrastructure (PKI).

- **Using your Private CA to sign your own SSL certificates:** You can become your own CA and create as many SSL certificates as you require. However, since your CA will not be included in the internal list of trusted CAs maintained by most browsers, customers will get a security warning until they add your CA to their browser, unless you manually add them to the browser's Trusted Root certificate store. As with self-signed certificates, these are intended for testing purposes only.

WARNING: You will not be able to purchase a certificate from a vendor for the purpose of 802.1x authentication, which in fact requires the SSL certificate be signed by a Trusted Root Certification Authority (CA), which they will no longer do. Instead, you would be expected to create your own PKI (Public Key Infrastructure) by using the OpenSSL tools & scripts, or some other tool, like Windows certificate server for managing your own PKI.

NOTE: As of January 1st, 2011 all certificate vendors will only provide SSL certificates are no longer signed by the "Trusted Root Authority" certificate, but are now signed by the vendor's Intermediate CA. This will mean creating a "certificate chain" which is described later in this document.

NOTE: Also, all certificate signing requests (CSRs) must now be generated with a 2048bit RSA key. The included certificate-creation scripts are already adjusted for this, and will generate CSRs with a 2048bit RSA key.

SSL Authentication and Trust Sequence

The following sequence of steps illustrates how an SSL session is established.

1. You associate your laptop to the MSM controller, and you get and IP address.

2. You open your web browser and attempt to reach your default URL.

3. The web server on the MSM intercepts this URL request, (because you're not yet authenticated) and redirects you to the secure login port, (port 8090).

4. The MSM Login Portal, (web server listening on port 8090) responds by sending its default certificate, as a way to establishing the secure SSL connection for the login, to your web browser, along with the login page.

5. The web browser then attempts to validate the web server's certificate authenticity to determine if it can be trusted, by comparing the "Issued by" field in the certificate and attempts to find a matching CA certificate (of the same name), in the browser's Trusted Root certificate store.

   The following criteria are checked during the certificate validation phase;

   • The web browser checks that the server's certificate has not expired. The certificate will contain the certificate's validity period which can be compared to the current date. The current date must be in the range of the certificate's validity period or a certificate warning is generated.

   • The web browser may be configured to check that the certificate is not in a Certificate Revocation List maintained by the certificate authority that signed the certificate. The certificate must not be listed in the revocation list or a certificate warning is generated.

   • The web browser checks its internal list of Trusted Root CA certificates to find the matching name of the "Issue by" field in the SSL certificate (that signed the SSL certificate being checked). Finding a trusted Root CA certificate name that matched the "Issued by" of the SSL certificate allows the web browser to validate the authenticity of the web server's SSL certificate. A matching CA certificate MUST be found in the browser's Trusted Root certificate store that can validate the SSL certificate identity or a certificate warning is generated.

   • The web browser compares the "Issue To" field of the SSL certificate from the MSM to the domain name in the URL being requested. They must match or a certificate warning is generated.

6. Once the web browser accepts the certificate as valid and trusted, the MSC and the browser can have an encrypted SSL connection.

7. The SSL connection is started. All data between the browser and web server is secure.

Installing OpenSSL for Windows

OpenSSL can be downloaded for free, and using the supplied scripts makes generating certificate easier. To use the supplied scripts, you will use OpenSSL for Windows;

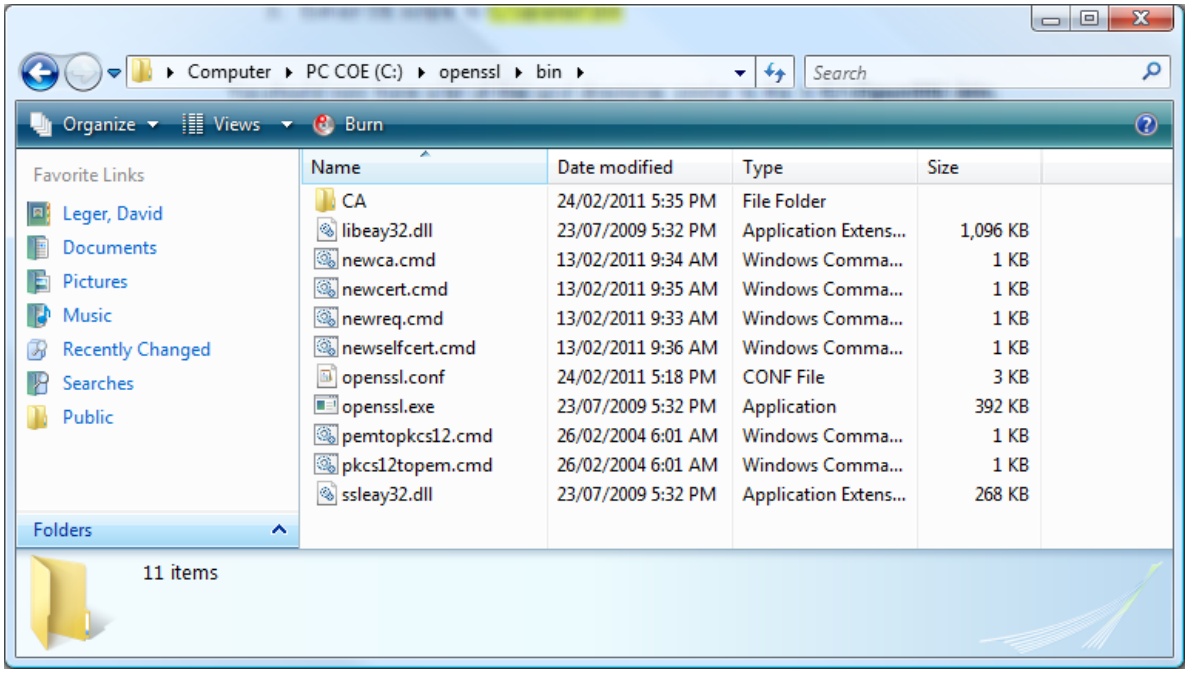
1. Select the latest version of OpenSSL and install it using its defaults. This document was validated using the **openssl-0.9.8k_WIN32.zip**. But the more recent versions can always be found at;

   http://www.deanlee.cn/programming/openssl-for-windows/

2. Extract the **openssl-0.9.8k_WIN32.zip** to the **C:\openssl** folder.

3. Extract **SSL scripts.zip** to the **C:\openssl\bin** folder.

You should now have a list of files and directories like this in **C:\OpenSSL\bin** folder;



Creating a Certificate Signing Request (CSR)

This example illustrates how to create a Certificate Signing Request and send it to a Certificate Authority to obtain a signed SSL certificate. The benefit of using a signed certificate is that the CA certificates from these Certificate Authorities are installed by default in most web browsers, thereby avoiding problems with certificate warning messages.

NOTE: You must specify a FQDN (fully qualified domain name) for the name of the "**Common Name**" field of the certificate request. If the certificate does not have an FQDN, it will not install properly on the MSM products.

NOTE: The FQDN that you choose **SHOULD NOT** be resolvable to a Public IP address.
It will have only local significance to clients that are assigned an IP address on the private DHCP subnet of the MSM. When your SSL certificate is installed on the MSM, the FQDN will automatically be resolved ONLY to the Lan port IP address of the MSM.

Certificate Naming: The MSM products only support standard FQDN naming convention, so;

Acceptable FQDN = hotspot.hp.com
Unacceptable FQDN = hp or ***.hp.com** or **192.100.50.25** (eg, an IP address)

For the purposes of this example, the certificate will be requested for the domain name: hotspot.hp.com, and the passphrase for the private key is 'my_password'.

1. Open a Windows command-line session. (For **Vista users**, you MUST have to right-click and select "**Run as Administrator**" to make OpenSSL work properly)

2. Go to the **c:\OpenSSL\bin** directory where you installed the certificate tools.

3. Execute the command: **newreq hotspot.hp.com**

(Everything you must enter is shown below in **Black** and the output is in **Orange**);

C:\OpenSSL\bin>**newreq hotspot.hp.com**
You will now be prompted for a password
that will protect the new private key.
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
Enter pass phrase: **my_password**
Verifying - Enter pass phrase: **my_password**
Re-enter the password for your new private key
(The same you just entered)
Enter pass phrase for hotspot.hp.com.key: **my_password**
You are about to be asked to enter information that will be incorporated
In to 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) [California]:**California**
Locality Name (eg, city) [Roseville]:**Roveville**
Organization Name (eg, company) [HP Networks]:**HP Networks**
Organizational Unit Name (eg, section) [Support]:**Support**
Common Name (Issued to: name) []: hotspot.hp.com
Email Address [support@hp.com]: **support@hp.com**
Generated certificate request:
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, ST=California, L=Roseville, O=HP Networks, OU= Support, CN=hotspot.hp
.com/emailAddress=support@hp.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:53:77:43:cc:ab:3c:e7:d4:c0:da:53:52:d8:
                    53:d8:ad:f9:93:ed:f1:f8:e6:9c:77:5e:3c:40:92:
                    f3:0a:3a:0a:19:5c:77:bd:c9:a3:fa:0a:13:66:61:
                    bc:b3:e8:cb:cb:ae:a5:81:7d:9a:4b:14:3f:28:f2:
                    5e:44:3c:01:7f:96:a6:32:7d:aa:0c:01:5e:f6:ba:
                    08:13:85:e8:c0:d3:8c:86:e1:99:c1:88:60:e2:fd:
                    d3:69:ee:82:b0:56:a7:c1:60:13:5b:fc:72:6d:28:
                    8d:93:36:f5:93:75:67:de:4b:8c:87:e4:bd:71:20:
                    30:f3:9e:98:99:2f:76:7d:dd
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha1WithRSAEncryption
        69:0d:e0:f1:62:4a:4a:95:4f:59:c7:1c:86:26:fd:08:03:be:
        ae:12:e4:2e:76:b2:cd:91:9d:79:fd:1d:2a:3d:0c:27:06:e7:
        83:b2:f6:d4:66:bc:57:d5:64:a9:64:ff:d3:75:f0:f1:9b:09:
        d9:39:1f:26:e8:8a:32:93:3e:fe:7c:41:80:7b:0b:3c:dd:eb:
        e4:78:f0:bb:7c:54:0d:3a:3c:59:a7:bf:33:02:d8:04:b0:ad:
        2e:a6:d2:64:de:c9:7d:ad:28:cb:fa:19:02:bc:7a:c3:3c:d9:
        97:43:da:e3:f6:77:85:1f:78:c6:73:f7:82:c6:60:aa:64:b8:
        0d:cc

-----BEGIN CERTIFICATE REQUEST-----
MIIB5jCCAU8CAQAwgaUxCzAJBgNVBAYTAkNBMQ8wDQYDVQQIEwZRdWViZWMxDjAM
BgNVBAcTBUxhdmFsMRowGAYDVQQKExFDb2x1YnJpcyBOZXR3b3JrczEaMBgGA1UE
CxMRVGVjaG5pY2FsIFN1cHBvcnQxGDAWBgNVBAMTD3d3dy5jb21wYW55LmNvbTEj
MCEGCSqGSIb3DQEJARYUc3VwcG9ydEBjb2x1YnJpcy5jb20wgZ8wDQYJKoZIhvcN
AQEBBQADgY0AMIGJAoGBANFTd0PMqzzn1MDaU1LYU9it+ZPt8fjmnHdePECS8wo6
Chlcd73Jo/oKE2ZhvLPoy8uupYF9mksUPyjyXkQ8AX+WpjJ9qgwBXva6CBOF6MDT
jIbhmcGIYOL902nugrBWp8FgE1v8cm0ojZM29ZN1Z95LjIfkvXEgMPOemJkvdn3d
AgMBAAGgADANBgkqhkiG9w0BAQUFAAOBgQBpDeDxYkpKlU9ZxxyGJv0IA76uEuQu
drLNkZ15/R0qPQwnBueDsvbUZrxX1WSpZP/TdfDxmwnZOR8m6loykz7+fEGAews8
3evkePC7fFQNOjxZp78zAtgEsK0uptJk3sl9rSjL+hkCvHrDPNmXQ9rj9neFH3jG
c/eCxmCqZLgNzA==
-----END CERTIFICATE REQUEST-----

C:\OpenSSL\bin>

At the end of this stage, two new files have been created in the **c:\OpenSSL\bin** folder;

hotspot.hp.com.key:
This file is the private key for the Certificate Signing Request.

KEEP THIS PROTECTED ! It's your private Key !

hotspot.hp.com.req:
The last section, (block of code shown above in Red in previous page), is the Certificate Signing Request, which you'll copy and send this to a Trusted Certificate Authority vendor, to have it signed by them.

(Each Certificate Authority has their own procedures, please consult with them).
It will be emailed back to you (or you will be instructed to download it) as a signed and trusted certificate which will be recognized by the majority of web browsers.

IMPORTANT: When asked by your certificate vendor, what type of web server the MSM uses, tell them the type is "**Apache**". The OpenSSL implementation uses the "**SSLeay**" library.

Creating your own CA Certificate

If you decide to be your own PKI and Certificate Authority, (CA), you can use this procedure to create your own CA certificates, which can then be later used to sign your own CSRs. This limits the effectiveness of the SSL certificate since it is signed by an unknown certificate authority.

This is what you would do when implementing 802.1x authentication, where clients by default will attempt to validate your SSL certificate, therefore you would need to install your CA certificate manually into their browser's Trusted Root Certificate store.

For the purposes of this example, the certificate will be requested for the domain name: **My-Company-CA**

1. Open a Windows command-line session. (Vista users MUST "**Run as Administrator**")

2. Go to the directory where you installed the certificate tools. This example assumes **c:\OpenSSL\bin** and pass-phrase is **my_password**.

3. Execute the command: **newca My-Company-CA**

(Everything you must enter is shown below in **Black** and the output is in **Orange**);

```
C:\OpenSSL\bin> newca My-Company-CA
*** You will be asked for a password protecting your ***
*** Certificate Authority Private Key ***
***
***
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'CAkey.pem'
Enter PEM pass phrase: my_password
Verifying - Enter PEM pass phrase: my_password
-----
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) [California]:California
Locality Name (eg, city) [Roseville]:Roseville
Organization Name (eg, company) [HP Networks]:HP Networks
Organizational Unit Name (eg, section) [Support]:Support
Common Name (Issued to: name) []:My-Company-CA
Email Address [support@hp.com]:support@hp.com
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            dc:a8:bd:88:d1:f6:7f:ff
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=California, L=Roseville, O=HP Networks, OU= Support, CN=My-Company-C
A/emailAddress=support@hps.com
```

```
        Validity
            Not Before: May 28 20:24:08 2007 GMT
            Not After : Dec  9 20:24:08 2027 GMT
        Subject: C=US, ST=California, L=Roseville, O=HP Networks, OU= Support, CN=My-Company-C
A/emailAddress=support@hps.com
Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:bc:a2:af:20:25:e1:90:c4:89:79:69:c6:30:4b:
                    d0:3d:c4:f2:0a:6f:83:ec:09:67:63:da:47:e8:9f:
                    a6:84:8c:3a:3d:57:22:1e:c1:d0:29:6d:c5:68:22:
                    03:65:5e:7f:cf:fc:ec:bb:ac:94:fb:cf:3a:53:aa:
                    a1:13:e9:8c:37:24:0c:df:a3:bb:0a:5f:47:89:d5:
                    ba:b8:49:b3:1d:99:da:af:ff:6d:c0:0d:b7:5c:17:
                    1e:bc:31:f4:60:fc:28:37:f1:80:fe:85:d5:6c:45:
                    16:83:dc:60:45:99:3f:dc:b6:63:a6:a0:c2:75:2e:
                    b9:61:e0:b3:33:e4:6f:45:f7:e1:11:d4:7b:c8:b2:
                    39:5d:99:46:e5:20:e7:50:2e:cf:2a:81:23:6a:91:
                    20:ff:cb:32:87:d9:9a:8a:92:80:11:7a:70:b3:53:
                    8f:98:da:c6:cd:60:03:a1:5d:2e:56:68:68:2b:ad:
                    d8:b5:c1:ad:c3:ce:8e:b2:b9:18:b4:45:87:b9:87:
                    fc:db
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
        b7:85:b6:b8:8c:37:e9:f9:e4:bb:a0:96:f9:eb:6b:44:50:c4:
        d9:8e:68:00:d8:7a:5d:7b:c2:df:37:66:fa:dc:99:a7:c0:0c:
        cf:a2:28:5e:43:db:15:4c:97:a5:d3:33:64:a5:18:25:4c:bc:
        7a:32:6d:d9:50:eb:af:61:6a:63:d8:8c:fc:60:89:a5:c2:26:
        15:2f:d4:94:f0:f9:5a:ff:82:64:57:3c:d2:db:e4:cd:b2:95:
        81:22:4b:d1:cb:d6:9a:94:28:3b:c0:19:bc:30:7e:01:b5:f0:
        c7:58:26:ce:60:a2:68:a5:32:12:37:38:12:50:78:f7:59:03:
        fd:c9:8e:6a:d4:04:8c:24:72:2a:d9:62:7f:9c:77:19:0d:1b:
        52:ca:b6:d4:66:20:97:2c:44:1a:97:c2:e7:03:3f:0c:f9:61:
        f1:e7:94:71:05:e4:66:58:5b:bb:43:92:34:1c:29:5d:45:2d:
        9f:62:1d:7d:6a:8b:07:e1:3f:0d:97:59:63:b9:42:e9:a0:27:
        4d:7e:43:0a:54:e8:d5:f1:ff:16:d6:ff:0a:a8:08:db:bd:76:
        a5:b5:2c:64

-----BEGIN CERTIFICATE-----
MIIDxDCCAqwCCQDcqL2I0fZ//zANBgkqhkiG9w0BAQUFADCBozELMAkGA1UEBhMC
Q0ExDzANBgNVBAgTBlF1ZWJlYzEOMAwGA1UEBxMFTGF2YWxvGjAYBgNVBAoTEUNv
bHVicmlzIE5ldHdvcmtzMRowGAYDVQQLExFUZWNobmljYWwgU3VwcG9ydDEWMBQG
A1UEAxMNTXktQ29tcGFueS1DQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9ydEBjb2x1
YnJpcy5jb20wHhcNMDcwNTI4MjAyNDA4WhcNMjcxMjA5MjAyNDA4WjCBozELMAkG
A1UEBhMCQ0ExDzANBgNVBAgTBlF1ZWJlYzEOMAwGA1UEBxMFTGF2YWxvGjAYBgNV
oq8gJeGQxIl5acYwS9A9xPIKb4PsCWdj2kfon6aEjDo9VyIewdApbcVoIgNlXn/P
/Oy7rJT7zzpTqh7zSR8DeUdztM/bLzyxXeNO+vPRRE450IzkeGNuRMc1ftMyG5+w
XwBwJWMJbUSc6GZh2npvqFbHhyx+/KET6Yw3JAzfo7sKX0eJ1bq4SbMdmdqv/23A
DbdcFx68MfRg/Cg38YD+hdVsRRaD3GBFmT/ctmOmoMJ1Lrlh4LMz5G9F9+ER1HvI
sjldmUblIOdQLs8qgSNqkSD/yzKH2ZqKkoARenCzU4+Y2sbNYAOhXS5WaGgrrdi1
wa3Dzo6yuRi0RYe5h/zbAgMBAAEwDQYJKoZIhvcNAQEFBQADggEBALeFtriMN+n5
5Luglvnra0RQxNmOaADYel17wt83ZvrcmafADM+iKF5D2xVMl6XTM2SlGCVMvPxn
+BTB1pBlSIclSe99zbJJbK6Wv1wUZLHYm5rJAKF8l1kgyHoybdlQ669hamPYjPxg
iaXCJhUv1JTw+Vr/gmRXPNLb5M2ylYEiS9HL1pqUKDvAGbwwfgG18MdYJs5gomil
MhI3OBJQePdZA/3JjmrUBIwkcirZYn+cdxkNG1LKttRmlJcsRBqXwucDPwz5YfHn
lHEF5GZYW7tDkjQcKV1FLZ9iHX1qiwfhPw2XWWO5QumgJ01+QwpU6NXx/xbW/wqo
CNu9dqW1LGQ=
-----END CERTIFICATE-----
C:\OpenSSL\bin>
```

At the end of this stage, two additional files have been created in the **c:\OpenSSL\bin folder**;

My-Company-CA.key: This is the private key for your CA certificate.

KEEP THIS PROTECTED ! It's your private Key !

My-Company-CA.pem: This is the CA certificate, (shown above in Red on the previous page). You can now use this CA certificate with the **newcert** command, which will generate CSR, then sign your own SSL certificates, in one process.

Note: If you take the **My-Company-CA.pem** file and rename the extension to **.crt**, and then you will be able to view it in Windows by double-clicking on it…

Note: The CA certificate created above, (**My-Company-CA**), needs to be installed in your browser's Trusted Root Certificate Store to be available so that the browser will accept any SSL certificate that you subsequently have signed by this CA certificate.

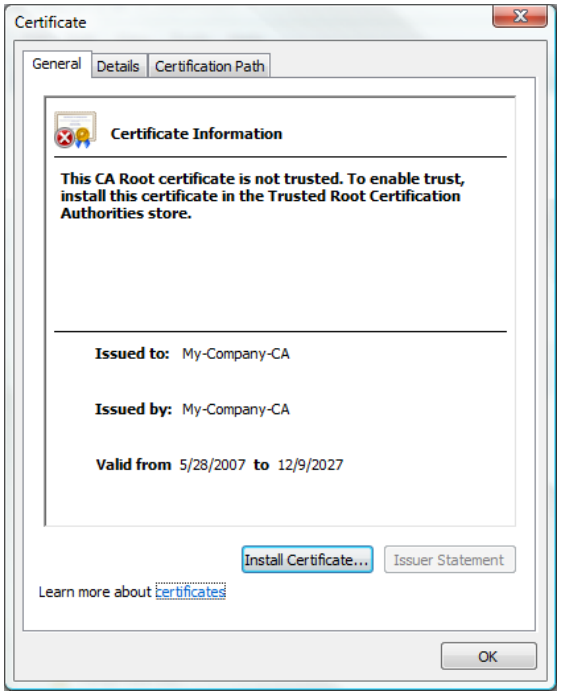
What does a certificate look like ?

When you've generated either your own CSR or SSL certificate with OpenSSL, it will be in "base-64" format, shown below in red. The can be copied into a text file and saved with either a .PEM or .CRT extension. Certificates always start with "**BEGIN CERTIFICATE**" and end with "**END CERTIFICATE**".

-----BEGIN CERTIFICATE-----
MIIDxDCCAqwCCQDcqL2I0fZ//zANBgkqhkiG9w0BAQUFADCBozELMAkGA1UEBhMC
Q0ExDzANBgNVBAgTBlF1ZWJlYzEOMAwGA1UEBxMFTGF2YWwxGjAYBgNVBAoTEUNv
bHVicmlzIE5ldHdvcmtzMRowGAYDVQQLExFUZWNobmljYWwgU3VwcG9ydDEWMBQG
A1UEAxMNTXktQ29tcGFueS1DQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9ydEBjb2x1
YnJpcy5jb20wHhcNMDcwNTI4MjAyNDA4WhcNMjcxMjA5MjAyNDA4WjCBozELMAkG
A1UEBhMCQ0ExDzANBgNVBAgTBlF1ZWJlYzEOMAwGA1UEBxMFTGF2YWwxGjAYBgNV
BAoTEUNvbHVicmlzIE5ldHdvcmtzMRowGAYDVQQLExFUZWNobmljYWwgU3VwcG9y
dDEWMBQGA1UEAxMNTXktQ29tcGFueS1DQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9y
dEBjb2x1YnJpcy5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC8
oq8gJeGQxIl5acYwS9A9xPIKb4PsCWdj2kfon6aEjDo9VyIewdApbcVoIgNlXn/P
/Oy7rJT7zzpTqh7zSR8DeUdztM/bLzyxXeNO+vPRRE450IzkeGNuRMc1ftMyG5+w
XwBwJWMJbUSc6GZh2npvqFbHhyx+/KET6Yw3JAzfo7sKX0eJ1bq4SbMdmdqv/23A
DbdcFx68MfRg/Cg38YD+hdVsRRaD3GBFmT/ctmOmoMJ1Lrlh4LMz5G9F9+ER1HvI
sjldmUbIlOdQLs8qgSNqkSD/yzKH2ZqKkoARenCzU4+Y2sbNYAOhXS5WaGgrrdi1
wa3Dzo6yuRi0RYe5h/zbAgMBAAEwDQYJKoZIhvcNAQEFBQADggEBALeFtriMN+n5
5Luglvnra0RQxNmOaADYel17wt83ZvrcmafADM+iKF5D2xVMl6XTM2SlGCVMvPxn
+BTB1pBlSIcISe99zbJJbK6Wv1wUZLHYm5rJAKF8l1kgyHoybdlQ669hamPYjPxg
iaXCJhUv1JTw+Vr/gmRXPNLb5M2ylYEiS9HL1pqUKDvAGbwwfgG18MdYJs5gomil
MhI3OBJQePdZA/3JjmrUBIwkcirZYn+cdxkNG1LKttRmlJcsRBqXwucDPwz5YfHn
lHEF5GZYW7tDkjQcKV1FLZ9iHX1qiwfhPw2XWWO5QumgJ01+QwpU6NXx/xbW/wqo
CNu9dqW1LGQ=
-----END CERTIFICATE-----

Likewise, the private keys used to generate requests are also in base-64 format and start with "**begin RSA Private key**" and end with "**end RSA Private key**"…

If a certificate is saved with a .crt extension, you can double-click on it Windows and view its details.



Creating an SSL certificate with your CA certificate

This procedure enables you to sign your own SSL certificates using your own CA certificate & private key. This is useful for testing or for 802.1x authentication.

Customers who have theiryour CA installed in their browsers will be able to trust the certificates you have signed. Same is true for 802.1x clients who must "validate server certificate".

You will be asked for a password to protect the new private key, which will be the private key for your own SSL Certificate.

For the purposes of this example, the **Certificate Authority** will be the name:

My-Company-CA (Previously created above)

1. Open a Windows command-line session.
2. Go to the directory where you installed the certificate tools. This example assumes **c:\OpenSSL\bin** and pass-phrase is you're **my_password**
3. Execute the command: **newcert www.company.com My-Company-CA**

(Everything you must enter is shown below in **Black** and the output is in **Orange**);

```
C:\OpenSSL\bin>newcert www.company.com My-Company-CA
*** Phase-1 Private Key generation          ***
*** You will now be prompted for a password ***
*** that will protect the new private key.  ***
***
***
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....
.....+++
e is 65537 (0x10001)
Enter pass phrase for www.company.com.key: my_password
Verifying - Enter pass phrase for www.company.com.key: my_password

*** Phase-2 CSR generation                  ***
*** Re-enter the password for your new private key ***
*** (The same you just entered)             ***
***
***
Enter pass phrase for www.company.com.key: my_password
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) [California]:California
Locality Name (eg, city) [Roseville]:Roseville
```

```
Organization Name (eg, company) [HP Networks]:HP Networks
Organizational Unit Name (eg, section) [Support]: Support
Common Name (Issued to: name) []:www.company.com
Email Address [support@hp.com]:support@hp.com

*** Phase-3  The "My-Company-CA" will now sign your "www.company.com" CSR        ***
*** You will now be prompted for the password for your ***
*** Certificate Authority private key.                  ***
***
***
Using configuration from openssl.conf
Loading 'screen' into random state - done
Enter pass phrase for My-Company-CA.key: my_password
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'California'
localityName          :PRINTABLE:'Roseville'
organizationName      :PRINTABLE:'HP Networks'
organizationalUnitName:PRINTABLE:'Support'
commonName            :PRINTABLE:'www.company.com'
emailAddress          :IA5STRING:'support@hp.com'
Certificate is to be certified until Dec 10 14:04:10 2027 GMT (7500 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
C:\OpenSSL\bin>
```

At this stage, two files have been created:

www.company.com.key, This is the private key for your SSL certificate

www.company.com.pem, This is your SSL certificate.

Note: If you take the **www.company.com.pem** file and rename it to **.crt**, you will be able to view the certificate in Windows by double-clicking on it…

Note: The CA certificate created above, (**My-Company-CA**), needs to be installed in your browser so that your browser's Trusted Root Certificate Store so it will accept the SSL certificate that you just created.

Creating a certificate chain

During the installation if the SSL certificate on the MSM controller, you must also supply the private key of the SSL certificate, in order for the MSM controller to be able to encrypt/decrypt the SSL traffic on its side. In order to do this, you must create a certificate chain containing at least the SSL certificate block and the private key block.

However, all SSL certificates purchased today are signed by an Intermediate CA certificate, which in turn is signed by the Trusted Root CA certificate. When a web browser connects to the MSM controller using SSL, the MSM only sends the installed SSL certificate to the browser initially.

If the web browser does not possess the Intermediate CA certificate for that SSL certificate in its certificate store, then the browser will most likely request it from the MSM controller. If the Intermediate CA certificate has not been also installed on the controller, (as part of the certificate chain), then the web browser will not get the whole certificate chain it needs to validate the identity of the SSL certificate and the client browser will see an SSL security warning..

The solution is to include all the components needed (**SSL Certificate, Intermediate CA** and **Private Key**) to satisfy the validity of your SSL certificate and for SSL to function properly. So, if your SSL certificate is signed by the Intermediate CA, then both certificates, plus the private key of your SSL certificate MUST be included in what is called a "certificate chain".

A certificate chain is nothing more than a text file that contains all the base-64 certificate blocks needed, plus the private key block of your SSL certificate, and then saved in a text format with a .pem extension.

NOTE: The Trusted Root certificate cannot be added to the chain, as it MUST exist in the client's Trusted Root certificate store already and cannot be supplied by the MSM controller. (The assumption here is that is that because it's already installed in the operating system's browser, then it is trustworthy, hence it can then validate other certificates signed by it.

Note: If you certificate is signed by an intermediate certificate authority, then you need to build a certificate chain so that the SSL cert, Intermediate cert and private key become part of the certificate chain file. Failing to do so will result is SSL failing for the clients.

Note: All cert components MUST be in base-64 format. (The key is already base-64)
A certificate in Base-64 format is any block of text that starts with "**BEGIN CERTIFICATE**" and ends with "**END CERTIFICATE**". If the content of your certificate file does not contain a block of text resembling this, then it may not be in Base-64 yet, and may need to be converted.

See the example on next page to see how a certificate chain is made…

We assume the following components will be added to the text file;

- The **www.company.com.pem** SSL certificate

- The **www.company.com.key** matching key file for your certificate

- The **Intermediate Authority** CA certificate, which signed your certificate.

Note: The Root CA certificate does NOT get added to the chain, but MUST exist in the user's browser(s) in order to complete the chain of validation, otherwise a certificate warning will occur.

Creation of your certificate chain inside an empty text file should look like this;
(colors are done for clarity and sequence is not important)

-----BEGIN RSA PRIVATE KEY-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx www.company.com.key xxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxx www.company.com.pem xxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxx Intermediate CA cert xxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----END CERTIFICATE-----

Save this as a plain text file, with a .pem extension, then follow the instructions below on how to convert this file to PCKS#12 format using the **PEMtoPKCS12.cmd** script before installing it on the MSM.

Converting a .PEM file to PKCS #12 format

Before you can install a certificate on the MSM, you need to convert it to PKCS #12 format. The PCKS#12 format provides a secure way of installing a certificate and its private key, without exposing an unprotected private key, and is a requirement for the MSM controllers. This can be done using the **PEMtoPKCS12.cmd** command file.

Execute the command:

PEMtoPKCS12 *CertificateName*

Replace *CertificateName* with the name of the certificate file, (without the PEM extension). Make sure that both the .pem and .key files are in the **c:\OpenSSL\bin** folder and have the same name (but with their respective different extensions, of course).

You will be prompted for two passwords:

PEM pass phrase: Password used to protect the private key (**my_password**)

Export password: Password that will lock the PKCS#12 file. You will specify this password when you load the certificate onto the MSM, (**my_password**).

For example:

C:\OpenSSL\bin>PEMtoPKCS12 www.company.com
Loading 'screen' into random state - done
Enter PEM pass phrase: **my_password**
Enter Export Password: **my_password**
Verifying password - Enter Export Password: **my_password**

This procedure will generate a file named **www.company.com.pkcs12.** This file should always contain both the private key and SSL certificate, (and the intermediate CA, if needed). This file can now be installed on the MSM.

Note: The "**my_password**" will then be required during the installation of the certificate into the MSM controller.

Installing the new SSL certificate onto the MSM

Before you can install the new SSL certificate, make sure that it conforms to the following requirements:

• It must be in **PKCS #12** format.

• It must contain the matching private key (and Intermediate CA if required).

• The "common name" must be a Fully Qualified Domain Name (FQDN), containing at least one dot. If you try to add a certificate with an invalid name, it will not load and the default certificate is restored.

NOTE: The FQDN that you choose SHOULD NOT be registered with INTERNIC or resolve to a specific public IP address. When your SSL certificate is installed on the MSM, the FQDN will automatically be resolved to the LAN port IP address of the MSM controller.

NOTE: SSL certificates with "wildcard domain names" are NOT supported on the MSM products. Only install certificates with fully qualified domain names, like "wireless.hp.com".

Acceptable FQDN = www.colubris.com
Unacceptable FQDN = **"colubris"** or **"*.colubris.com"** or **"192.168.100.25"**

Where can I buy an SSL certificate ?

Certificates can be purchased from a variety of Root and Intermediate certificate vendors. Their sites are generally full of additional reading materials on the subject.
Here are some SSL certificate vendors;

http://www.entrust.com

http://www.verisign.com

http://www.thawte.com

http://www.godaddy.com

Here are some SSL certificate information resources;

http://www.whichssl.com/faqs/browser.html

http://www.openssl.org/docs/

http://en.wikipedia.org/wiki/Public_key_certificate

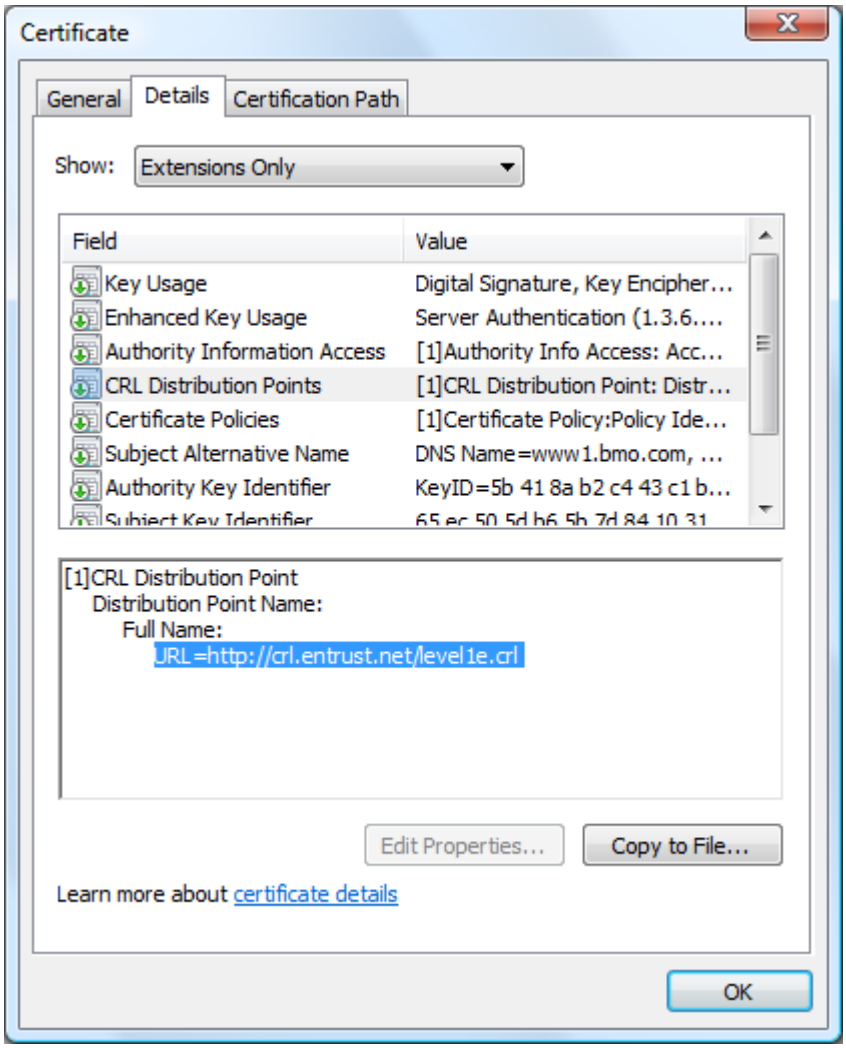
Certificate Revocation List (CRL)

If you replaced the default SSL certificate on the controller with one signed by a well known CA, you should define the access list to permit access to the certificate vendor's revocation list, posted on their website, too allow all non-authenticated users to validate that the SSL certificate they have just received has not be revoked.

This enables the user's browser to verify that the certificate is valid without displaying any warning messages. Users may have configured their Web browsers to check all SSL certificates against the Certificate Revocation List (CRL) maintained by the vendor that issued the certificate. The location of the CRL may be configured in the browser, or in general, it more conveniently embedded in the certificate as a URL link. The access list should be configured to permit access to the CRL, otherwise the user's browser may time out before displaying the login page, while unsuccessfully attempting to retrieve the revocation list.

You can find out what the URL for the vendor's CRL by inspecting the SSL certificate itself. Once you've installed the SSL certificate in the controller, then you should be able to launch your browser, get the login page in secure session, then click on the "padlock" in your browser to view the certificate.

Inside the certificate is where you'll see the Vendor's CRL URL link. It is this URL that would add to the controller's ACL (access control list); (i.e. access-list=ACCEPT,all,crl.entrust.net,all)



Share with colleagues

4AA2-xxxxENW, Created Month 2011