



Using Certificates in HTTPS Clusters

The sections below tell you how to get your Layer 7 HTTPS clusters running with certificates. Please read these sections completely before beginning to work with certificates on Equalizer.

While this document tells you all you need to know to use certificates with HTTPS clusters, it is *not* a primer on HTTPS, SSL, or certificates. There are many resources on the Internet, in trade publications, and in books on these topics. Most SSL certificate vendors offer basic SSL overviews on their websites.

Using Certificates in HTTPS Clusters	192
About Layer 7 HTTPS Clusters	192
About Certificates	192
Software vs. Hardware Encryption/Decryption	193
Using Certificates in a Failover Configuration	193
Enabling HTTPS with a Server Certificate	194
Enabling HTTPS with Server and Client Certificates	194
Generating a CSR and Getting It Signed by a CA	195
Generating a CSR using OpenSSL	196
Generating a Self-Signed Certificate	196
Preparing a Signed CA Certificate for Installation	197
Installing Certificates for an HTTPS Cluster	198
Using IIS with Equalizer	200
Generating a CSR and Installing a Certificate on Windows Using IIS	200
Converting a Certificate from PEM to PKCS12 Format	201
Private Key Storage for Cluster Certificates	202
Clearing Secure Key Storage	202
Configuring Cipher Suites	203
Default Cipher Suites	203
Updating the Cipher Suites Field	203
No Xcel and Xcel II Card Cipher Suites	203
Xcel I Card Cipher Suites	204

Using Certificates in HTTPS Clusters

The HTTPS protocol supports encrypted, secure communication between clients and servers. It requires that a Secure Sockets Layer (SSL) authentication handshake occur between a client and a server in order for a connection request to succeed.

When a client requests an HTTPS connection to a web server, the server (which has already been set up to support SSL connections) sends a *server certificate* to the client for verification. The client checks the content of the certificate against a local database of *Certificate Authorities*, and if it finds a match the connection is made. If no match is found (as is often the case with self-signed certificates), the browser will display a warning and ask if you want to continue with the connection.

A further level of trust can be enabled by setting the server up to request a *client certificate* in addition to the server certificate. Copies of the client certificate are pre-installed on both client and server. When the server sends the server certificate to the client, it also sends a request for a certificate from the client. Once the client accepts the server certificate as described above, it sends the client certificate to the server for verification. The server compares the client certificate it receives with its local copy of the client certificate, and if they match the connection is made.

A server certificate is required for an HTTPS connection; a client certificate is optional.

About Layer 7 HTTPS Clusters

In the typical HTTPS scenario described above, the client and server are communicating directly, and the server is doing all the work of encrypting and decrypting packets, comparing certificates, and authenticating clients. If you have many systems servicing requests for the same website, you'll need to install certificates on each server.

With Equalizer, you do not need to install a certificate on every server in a Layer 7 HTTPS cluster. Since certificates are associated with host names and not IP addresses, you only need a server certificate for each HTTPS cluster and the certificates are installed only on Equalizer -- not on each server. This reduces maintenance by reducing the number of certificates required for a group of systems serving content for the same host name.

When a client requests a connection to an HTTPS cluster, Equalizer establishes the HTTPS connection with the client, off loading SSL processing from all the servers in the HTTPS cluster. Equalizer communicates with the clients via HTTPS; the traffic between Equalizer and the servers in an HTTPS cluster is HTTP (i.e., unencrypted). Compared to the typical scenario where each server is establishing direct HTTPS connections with clients, encrypting and decrypting packets, and serving content as well, SSL offloading improves the overall performance of the cluster.

For even better performance, an optional Xcel SSL Acceleration Card can be installed in Equalizer. With Xcel, all SSL processing is done by the Xcel card, enhancing overall HTTPS throughput. For more information on Xcel, please visit the Coyote Point website (www.coyotepoint.com).

Note that HTTPS and certificates can be used on servers in Layer 4 TCP and UDP clusters, but you *will* need to install a server and client certificate on *each* server in the cluster (since Equalizer is not doing any HTTPS/SSL processing in Layer 4). In this scenario, no certificates are installed on Equalizer. Using a Layer 4 cluster is the preferred method for passing HTTPS traffic through Equalizer when you do not need to take advantage of features that are specific to Layer 7, such as cookie persistence, match rules, etc.

About Certificates

Each Layer 7 HTTPS cluster requires a *server* certificate; a *client* certificate is optional.

Web servers (such as Apache) and browsers (such as Internet Explorer and Firefox) are delivered with pre-installed Trusted Root Certificates. Trusted Root Certificates are used to validate the server and client certificates that are exchanged when an HTTPS connection is established.

Equalizer supports self-signed certificates, as well as signed certificates from Trusted Root Certificate Authorities and from Certificate Authorities (CAs) without their own Trusted Root CA certificates. If a CA without its own Trusted Root CA certificate issues your certificate, you will need to install at least two certificates: a server certificate and a chained root (or intermediate) certificate for the CA. The intermediate certificate associates the server certificate with a Trusted Root certificate.

Similarly, if you want to use client certificates with an HTTPS cluster, you'll need to get a signed client certificate from a CA, or create a self-signed certificate. A client certificate needs to be installed on each client that will access the Equalizer cluster, as well as on Equalizer. The same client certificate can be used on all clients (i.e., you don't need to buy or create a separate certificate for each client system).

Just as with server certificates, you may need to install a client certificate and a chained root certificate, if you obtain your certificates from a CA without its own Trusted Root CA certificate. Some sites prefer to use self-signed certificates for clients, or set up their own local CA to issue client certificates.

For several good tutorials on how to get your certificates signed, please see:

<http://sial.org/howto/openssl/>

Whichever method you choose, follow these general guidelines for certificates you want to use with Equalizer:

- Equalizer accepts both the **x509 PEM** or **PKCS12** certificate formats; PEM files usually have a *.pem* extension; PKCS12 files usually have a *.pfx* extension. Most CA vendors provide certificates in PEM format.
- If you are using an Xcel I accelerator card, use a private key **bit length** that is a multiple of **8** (e.g., 1024, 2048, etc.). This restriction does not apply to newer generation Xcel II cards.
- When uploading certificates to Equalizer, the certificates and private key must be contained in a single plain text file, in the following order:
 - server certificate
 - private key
 - chained root (intermediate) certificates (if any)

Software vs. Hardware Encryption/Decryption

Lower-end Equalizer models (the E250 and E350) perform all Layer 7 HTTPS encryption/decryption in software, using Equalizer's CPU and memory. Higher-end Equalizer models (the E450 and E650) provide hardware based encryption/decryption on an Xcel SSL Acceleration add-on card. With Xcel, all SSL operations for Layer 7 HTTPS clusters are performed on the card, thus offloading both the servers behind Equalizer and Equalizer itself -- freeing more resources for traffic and application management.

In terms of configuration, both software and hardware SSL operations require a list of cipher suites (encryption algorithms) to be used to encrypt and decrypt HTTPS traffic. The supported cipher suites for each SSL processing mode (software, Xcel I, Xcel II) are described in the section "Configuring Cipher Suites" on page 203.

Also see the section "Private Key Storage for Cluster Certificates" on page 202 for a discussion of how Equalizer stores the private keys for your cluster certificates, and keeping private keys secure on Equalizer.

Using Certificates in a Failover Configuration

In failover configurations, client and server certificates are *not* part of the configuration settings that are transferred between the failover peers when configuration changes are made on one of the failover systems. For this reason, you must install the server certificates (and the client certificates, if used) on *both* of the failover peers.

Enabling HTTPS with a Server Certificate

The following are the steps to follow to obtain and install a server certificate, and verify that it works.

1. Generate a Server Certificate Signing Request or a Self-Signed Server Certificate.

To get a server certificate, do *one* of the following:

- a. **Create a Certificate Signing Request (CSR) and send it to a Certificate Authority for signing.** This provides the highest level of trust to the client, as the client can be assured that the certificate it receives from the server (in this case, Equalizer) was approved (i.e., digitally signed) by a trusted third party. Thus, the client has the assurance of a third party that the server to which it is connecting is identifying itself legitimately (and is not impersonating the legitimate server's identity). See the section "Generating a CSR and Getting It Signed by a CA" on page 195.
 - b. **Create a certificate and sign it yourself.** This provides a lower level of trust, since the client is essentially trusting the server to identify itself. Self-signed certificates are relatively easy to counterfeit, and are only recommended for use on internal, non-production, or test configurations. See the section "Generating a Self-Signed Certificate" on page 196.
2. Create the HTTPS cluster.

When creating an HTTPS cluster, the default flags and parameters are acceptable for most server certificate configurations. However, if the server certificate you have does not strictly conform to the standard x509 format, disable the **x509 verify** flag in the cluster options. Many self-signed and some chained certificates may not be in strict x509 format.

For more information on SSL parameters, see the section "Layer 7 SSL Tab (HTTPS only)" on page 75.

3. Install the Server Certificate on Equalizer.

Use the Equalizer Administration Interface to install the server certificate. See the section "Installing Certificates for an HTTPS Cluster" on page 198.

4. Try connecting to the Cluster via HTTPS.

From a client browser, open **https://cluster**, where *cluster* is the network node name or IP address of the HTTPS cluster. The browser may notify you that it is accepting a certificate from the server and ask for confirmation. Once you accept the certificate, the requested page should be displayed.

Enabling HTTPS with Server and Client Certificates

The following are the steps to follow to obtain and install both server and client certificates, and verify that they work.

1. Perform the procedure in the previous section ("Enabling HTTPS with a Server Certificate" on page 194) to enable HTTPS with a server side certificate.
2. Generate a Client Certificate Signing Request or a Self-Signed Client Certificate.

In Step 1, you created a server certificate. Now, follow the same procedure to generate a client certificate; do *one* of the following:

- a. **Create a Certificate Signing Request (CSR) and send it to a Certificate Authority for signing.** See the section "Generating a CSR and Getting It Signed by a CA" on page 195.
- b. **Create a certificate and sign it yourself.** See the section "Generating a Self-Signed Certificate" on page 196.

Many organizations choose to use third-party signed certificates for their HTTPS clusters, and use self-signed certificates for their clients.

3. Modify the HTTPS cluster to request a client certificate.
 - a. Select the HTTPS cluster in the left frame of the Equalizer Administrative Interface and then select the **SSL** tab in the right frame.
 - b. Enable the **certify_client** flag; this tells Equalizer to request a client certificate when a client attempts to connect to this cluster.
 - c. By default, the **client certificate verification depth** is set to 2. This number indicates the number of levels in a certificate chain that the Equalizer will process before stopping (and refusing the connection). This default will need to be raised if you received more than one chained root certificate in addition to a client certificate from your Certificate Authority. Note that this setting has an impact on performance, since SSL operations are resource intensive.
 - d. By default, Equalizer requests a client certificate, but does not *require* the client to provide one. Enable the **require certificate** flag to require that a client return a valid certificate before connecting.
 - e. By default, the client's certificate will be re-validated if the SSL connection needs to be renegotiated. (Renegotiation is a feature of SSL, can occur for any of a number of reasons, and may be initiated by Equalizer or the client browser.) Enable the **verify once** flag to tell Equalizer *not* to re-evaluate the client certificate even if SSL renegotiation occurs. This can have a positive performance impact if many SSL renegotiations are occurring during normal operations.
 - f. Select **commit** to save your changes to the cluster definition.

For more information on SSL parameters, see the section “Layer 7 SSL Tab (HTTPS only)” on page 75.

4. Install the Client Certificate on Equalizer.

Use the Equalizer Administration Interface to install the client certificate. See the section “Installing Certificates for an HTTPS Cluster” on page 198.

5. Install the Client Certificate on all clients.

Import the client certificate into the client browser's list of certificates. On Firefox, open **Tools > Options > Advanced > View Certificates**. On Internet Explorer, open **Tools > Internet Options > Content > Certificates**. Refer to the documentation for your browser for instructions.

6. Try connecting to the Cluster via HTTPS.

From a client browser, open **https://cluster**, where *cluster* is the network node name or IP address of the HTTPS cluster. The browser may notify you that it is accepting a certificate from the server and ask for confirmation. Once you accept the certificate, the server should ask for a client certificate; your browser may ask you to choose one. After the client certificate is sent to the server and accepted, the requested page should be displayed.

Generating a CSR and Getting It Signed by a CA

Most CA vendors provide a means of generating a Certificate Signing Request (CSR) on their websites, and we recommend that you use the CA website to generate the CSR.

A CSR can also be generated using the OpenSSL tools on any system, including Windows. The examples below were executed on a Windows system with the OpenSSL tools installed.

Note that only the most basic **openssl** command options are shown. See the **openssl(1)** and **req(1)** manual pages at <http://www.freebsd.org/cgi/man.cgi> for more information. Many certificate vendors also provide tools on their websites for entering a CSR.

Generating a CSR using OpenSSL

1. Navigate to an appropriate directory on your system, and create a new directory to hold your CSR, certificate, and private key.
2. Generate the CSR by entering this command:

```
openssl req -new -newkey rsa:1024 -out cert.csr
```

This begins an interactive session to generate a CSR, and also generates a new private key to be output into a file named *privkey.pem*. The key length you use (1024 in this example) can be any multiple of 8. If you already have a private key, use **-key filename** (instead of **-newkey rsa:1024**) to specify the file containing the private key. The key length you use (i.e., 1024 in this example) can be any multiple of 8.

After generating the private key, the following prompts are displayed (example responses shown):

```
Enter PEM pass phrase: <password>
Verifying - Enter PEM pass phrase: <password>
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:Millerton
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CPS Inc.
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:mycluster.example.com
Email Address []:admin@example.com
```

Make sure you remember the **password** you specify, as you will need it to install and use the certificate.

For a *server certificate*, the **Common Name** provided must be the DNS-resolvable fully qualified domain name (FQDN) used by the Equalizer cluster. When a client receives the certificate from the server, the client browser will display a warning if the **Common Name** does not match the hostname of the request URI.

For a *client certificate*, the **Common Name** in the client's copy of the certificate is only compared to the **Common Name** in the copy of the client certificate on the server, so **Common Name** can be any value.

3. Visit the website of an SSL Certificate Authority (CA) to submit the *cert.csr* file to the CA.
4. Once the CA returns your signed certificate (usually in email), go to the section "Preparing a Signed CA Certificate for Installation" on page 197.

Generating a Self-Signed Certificate

To generate a self signed certificate in PEM format:

1. Generate a self-signed x509 format certificate by entering this command:

```
openssl req -new -x509 -newkey rsa:1024 -out selfcert.pem -days 1095
```

This creates a self-signed certificate (*selfcert.pem*) that will be valid for 1095 days (about three years) and also generates a new private key to be output into a file named *privkey.pem*. The key length you use (1024 in this example) can be any multiple of 8. If you already have a private key, use **-key filename** instead of **-newkey rsa:1024** to specify the file containing the private key. The key length you use (i.e., 1024 in this example) can be any multiple of 8.

After generating the private key, the following prompts are displayed (example responses shown):

```
Enter PEM pass phrase: <password>
Verifying - Enter PEM pass phrase: <password>
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:Millerton
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CPS Inc.
```

```
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:myclient.example.com
Email Address []:admin@example.com
```

Depending on the tool you use to create the certificate, you may also be asked for a challenge password and other optional information. Make sure you remember the **password** (and, if prompted, the challenge password) you specify, as you will need it to install the certificate.

The **Common Name** provided must be the DNS-resolvable fully qualified domain name (FQDN) used by the Equalizer cluster. For a *server certificate*, when the client receives the certificate from the server, the browser will display a warning if the **Common Name** does not match the hostname of the request URI. For a *client certificate*, the **Common Name** in the client's copy of the certificate is only compared to the **Common Name** in the copy on the server, so this can be any value.

2. Combine the private key and certificate into one file, using a command like the following:

```
cat selfcert.pem privkey.pem > clustercert.pem
```

3. You can now install your self signed certificate and private key file, *clustercert.pem*, on Equalizer and your clients, as appropriate.

Preparing a Signed CA Certificate for Installation

When you receive your signed certificate back from your CA, you'll get one or more *.pem* files in return, or you'll get one or more mail messages from the CA. The files or messages contain your signed certificate and any necessary intermediate certificates required by the CA's chain of trust.

If you get your certificates in the mail, save each one to an ASCII text file with a *.pem* extension. Make sure you use a text editor such as **Notepad** (Windows) or **vi** (Unix/Linux) to save the files as text files.

Note that if you are using IIS, see the section "Using IIS with Equalizer" on page 200.

If you get only *one* certificate (the signed server certificate) from your CA, then:

1. Save it to a text file (e.g., *servcert.pem* for a server certificate, or *clientcert.pem* for a client certificate).
2. Open a new text file and read both the signed certificate and your private key (in this order) into the file. (The private key was created previously when you generated your CSR.) Save the file as a plain text file. On a Unix system, like Equalizer, you can do this with a command like one the following:

```
cat servcert.pem privkey.pem > clustercert.pem
cat clientcert.pem privkey.pem > clientprivcert.pem
```

Whatever method you use, the file should look like this when you are done:

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
```

Make sure you save the file as a plain text file.

3. Install the file into Equalizer as instructed in the section "Installing Certificates for an HTTPS Cluster" on page 198.

If the CA uses chained root, or intermediate, certificates, then you'll receive (or need to download from the CA) more than one *.pem* file: the server certificate, plus any intermediate certificates needed to establish the chain of trust back to a Root CA certificate installed on your web server or client browser.

If you get *more than one* certificate (the signed server certificate plus one or more intermediate certificates) from your CA, then:

1. Save each certificate to a separate text file (e.g., *servcert.pem*, *intmcert.pem*).
2. Open a new text file and read the signed certificate, your private key, and any intermediate certificates (in this order) into the file. (Your private key was created previously, when you generated the CSR.) Save the file as a plain text file. On a Unix system, like Equalizer, you can do this with a command like one of the following:

```
cat servcert.pem privkey.pem intmcert.pem > clustercert.pem
cat clientcert.pem privkey.pem intmcert.pem > clientprivcert.pem
```

Whatever method you use, the file should look like this when you are done:

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
Add more certificates here if needed in the chain...
```

Make sure you save the file as a plain text file.

3. Install the file into Equalizer as instructed in the section “Installing Certificates for an HTTPS Cluster” on page 198.

Installing Certificates for an HTTPS Cluster

Your certificate authority may issue you either a single signed client or server certificate, or a signed certificate plus one or more chained root certificates (also called “intermediate” certificates). The certificate or certificates you receive establish a chain of trust that ends at a trusted root certificate installed on your web server (and on every client that interacts with the web server).

You must install all the certificates you receive on Equalizer to complete the installation process for HTTPS clusters. To install them on Equalizer, certificates must be in a single file, in either x509 (*.pem*) or PKCS12 (*.pfx*) format; see the section “Preparing a Signed CA Certificate for Installation” on page 197.

Caution – The private key for your *server* certificate is kept on Equalizer (in the directory */var/eq/ssl*) and will be accessible to anyone who can log into Equalizer. It is therefore essential that you restrict the ability of non-authorized personnel to access Equalizer, since any user can log in and copy or remove your private key. All Equalizer logins should be password protected with non-trivial passwords to restrict access to your private keys, and passwords should be given only to trusted personnel. Note that the private key for a *client* certificate (if used) is not stored on Equalizer, only the client certificate.

To install certificates onto Equalizer, follow these steps:

1. Copy the file containing the certificate and private key information (*clustercert.pem* in the examples above; *clustercert.pfx* if you used IIS) to the machine from which you will log into the Equalizer Administrative Interface. Note the location.
2. Log into the Administrative Interface using a login that has **add/del** access on the cluster that requires the certificate (see “Logging In” on page 33).

3. In the left frame, click the name of the HTTPS or SSL cluster for which you want to install a certificate and select the **Security > Certificates** tab in the right frame:

The screenshot shows two sections of the Certificates tab interface:

- select client or cluster certificate:** This section contains a red instruction: "For client verification, upload a single client certificate to authenticate all clients. For server verification, upload a single server certificate for the cluster." Below this are two radio buttons: "client" (unselected) and "cluster" (selected).
- select SSL certificate file:** This section contains a red instruction: "The certificate file must be in PEM (.pem) or PKCS12 (.pfx) format, and must contain the private key and the entire certificate chain." Below this is a text input field, a "Browse..." button, and an "upload" button.

Figure 61 The cluster Certificates tab

4. If your Equalizer has an **Xcel I** SSL Accelerator Card installed, a check box labeled **use secure key storage** will appear at the top of the **select client or cluster certificate** field. Checking this box tells Equalizer to store your private key in write-only memory on the Xcel card so that no one can access it. See the section "Private Key Storage for Cluster Certificates" on page 202, for more information.
If you have an Xcel II Card, or no Xcel Card, then this option will not appear on the screen; go to the next step. (Xcel II does not support storing private keys on the card.)
5. If you are installing a *server* certificate, leave the **cluster** radio button selected; if you are installing a *client* certificate, make sure that the **client** radio button is selected.
6. Enter the full path name of the certificate file (or click **Browse** to select the file). Click **upload** to install the certificate on Equalizer. You'll be prompted for a password, which is the password (PEM pass phrase) you provided when you generated the CSR for the certificate (or created the self-signed certificate).

Note – Uploading the certificate can fail for a number of reasons. For example, if the **x509 verify** cluster flag is enabled, Certain self-signed or chained certificates will not pass this verification. If you have trouble uploading your certificate, you may need to disable the **x509 verify** cluster flag and restart this procedure. See the description of "**x509 verify**" on page 75.

After the upload is complete, the **Certificates** tab displays the certificate details (serial number, key length, etc.) at the bottom of the tab.

Additional SSL settings, including the cipher suites permitted, appear on the **SSL** tab. See "Layer 7 SSL Tab (HTTPS only)" on page 75 and "Default Cipher Suites" on page 203 for more information.

7. If the certificate you just installed on Equalizer is a client certificate, you'll also need to install the certificate on each client. This usually involves converting the PEM format certificate into PKCS12 format; see the section "Converting a Certificate from PEM to PKCS12 Format" on page 201.

Using IIS with Equalizer

Using Internet Information Services (IIS) is optional when creating and managing certificates for Equalizer Layer 7 HTTPS clusters and clients. In fact, one of the advantages of using Equalizer is that only one server certificate is required for an HTTPS cluster. The cluster certificate is installed on Equalizer, *not* on the servers in the HTTPS cluster. So, you do not need to use IIS on each server to create and install certificates. This reduces the amount of effort spent administering server certificates.

For Layer 4 TCP and UDP clusters, certificates are *not* installed on Equalizer, and you *will* need to install a server certificate on *each* server in the cluster (since Equalizer is not doing any HTTPS/SSL processing in Layer 4). Generating a CSR and installing a signed certificate on Windows using IIS is shown in the procedure below.

Note that IIS does not support the creation of self-signed certificates. You must create the self-signed certificate on Equalizer (see “Generating a Self-Signed Certificate” on page 196) or another system that supports the OpenSSL tools; then, use IIS to import the certificate into the proper certificate store (usually, the **Personal** store) on Windows.

For more information on using IIS, please refer to the IIS documentation from Microsoft.

Generating a CSR and Installing a Certificate on Windows Using IIS

1. If you have not already installed Internet Information Services (IIS), use the **Add and Remove Programs** wizard (under **Control Panel**) to install it. Click on **Add/Remove Windows Components** and turn on the check box next to **Internet Information Services (IIS)**; click **Next** and follow the wizard’s instructions.
2. Select **Control Panel > Administrative Tools > Internet Information Services**.
3. For a cluster (server) certificate, navigate to the website for which the CSR is intended. For a client certificate, navigate to any website or the default. Right click on the website and select **Properties**.
4. Select the **Directory Security** tab and click the **Server Certificate** button.
5. Select **Next**, and follow the Certificate Wizard prompts:
 - a. Select **Create a new certificate**, and then **Next**.
 - b. Select **Prepare the request now, but send it later**, and then **Next**.
 - c. Type a **Name** for the certificate and select a **Bit Length** that is a multiple of 8. For most purposes, a bit length of 1024 is adequate. Longer bit lengths increase security at the expense of more SSL processing. Select **Next**.
 - d. Type in an **Organization** (e.g., **MyCompany, Inc.**) and **Organizational Unit** (e.g., **Marketing**); then select **Next**.
 - e. Type in the **Common name** for the certificate, and then select **Next**.

For a *server certificate*, the **Common Name** provided must be the DNS-resolvable fully qualified domain name (FQDN) used by the Equalizer cluster. When a client receives the certificate from the server, the client browser will display a warning if the **Common Name** does not match the hostname of the request URI.

For a *client certificate*, the **Common Name** in the client’s copy of the certificate is only compared to the **Common Name** in the copy of the client certificate on the server, so **Common Name** can be any value.

- f. Type in a **Country/Region**, **State/province**, and **City/locality**; then select **Next**.
- g. The last step in the wizard is to name and locate the new CSR. The default name and location will be `c:\certreq.txt` unless you choose otherwise.
6. Visit the SSL vendor’s website to submit your certificate request.

7. Once the SSL vendor has mailed the new signed certificate back to you, do one of the following:
 - a. If you are using this certificate with a Layer 4 cluster, copy the new certificate onto the system on which you generated the request and double-click to install. If this is a server certificate for a server in a Layer 4 TCP or UDP cluster, make sure you attach it to the appropriate web site. If this is a client certificate, make sure you place the certificate in the **Personal** certificate store.
 - b. If you are using the certificate with a Layer 7 cluster, export your new SSL certificate with your private key, so that it can be installed on Equalizer:
 - a. In IIS, right click on the website for which the certificate was generated and navigate through **Properties > Directory Security > View Certificate > Details**.
 - b. Select **Copy to File**, then **Next**.
 - c. Select **Yes**, export the private key; then **Next**.
 - d. Select **PKCS #12 (.PFX)**; check **Enable strong protection**; then **Next**.
 - e. Type and confirm the password; then **Next**.
 - f. Enter a file name, e.g. *C:\clustercert.pfx*; then click **Next**.
 - g. Click **Finish**.
 - h. Click **Ok** if the export was successful.
 - i. The certificate is now ready to be uploaded to the cluster via the Equalizer Administration Interface; see “Installing Certificates for an HTTPS Cluster” on page 198.

Converting a Certificate from PEM to PKCS12 Format

Many browsers, such as FireFox and Internet Explorer, require private keys and certificates in PKCS12 format for installation. In order to install client and intermediate certificates into these browsers, you will first have to convert them from PEM format to PKCS12 format. (Note: if you created your certificate using IIS as explained in the previous section, then your certificate is already in PKCS12 format; it can be installed directly into a browser without conversion.)

Like PEM format, PKCS12 format supports having all your certificates and your private key in one file, as discussed above in the section “Preparing a Signed CA Certificate for Installation” on page 197. If you followed the instructions in that section and created the file *clientprivcert.pem* (containing the client certificate, the private key, and any intermediate certificates), then converting the file to PKCS12 is simple:

```
openssl pkcs12 -export -in clientprivcert.pem -out clientprivcert.pfx
```

The resulting file, *clientprivcert.pfx*, can now be installed into all client browsers that will be accessing the cluster that requires a client certificate.

In **Internet Explorer**, certificates are installed by selecting **Tools > Internet Options** from the main menu, selecting the **Content** tab, and pressing the **Certificates** button. Select the **Personal** tab and then the **Import** button.

In **FireFox**, certificates are installed by selecting **Tools > Options** from the main menu, selecting **Advanced**, selecting the **Encryption** tab, and pressing the **View Certificates** button. When the **Certificate Manager** appears, select the **Your Certificates** tab and then the **Import** button.

Private Key Storage for Cluster Certificates

When you upload a *cluster* certificate to Equalizer, the uploaded file contains:

- the cluster certificate
- zero or more intermediate certificates
- the private key for the cluster certificate (chosen by you when you created the certificate signing request or self-signed certificate)

The private key should be guarded carefully and access to it restricted to those who administer Equalizer. If you do not have an Xcel card, or if you have an Xcel II card, private keys are kept in Equalizer's file system. The Xcel I card also provides the option to store private keys in memory on the Xcel I card.

Note that you should *not* check the **sks** check box when uploading *client* certificates, which are always stored on Equalizer *without* a private key.

The Equalizer Xcel SSL Accelerator Card is an add-on for Equalizer that provides hardware-based SSL encryption and decryption. There are two versions of the Xcel card, Xcel I and Xcel II. The older version, Xcel I, optionally supports storing private keys for cluster certificates in write-only memory on the Xcel card -- this is called **secure key storage** (SKS). All private keys uploaded to write-only memory can only be accessed by the accelerator hardware, thus preventing unauthorized access to your private keys.

If your Equalizer has an Xcel I SSL Accelerator Card installed, a check box labeled **use secure key storage** will appear on an HTTPS cluster's **Certificates** tab (see Figure 61). Checking this box tells Equalizer to store your private key in write-only memory on the Xcel card so that no one can access it.

The Xcel I card provides 128 kilobits of memory for private keys. This will hold up to 128 one-kilobit (1024-bit) keys, the key length supported by the Xcel I card. Be sure to use only 1024-bit private keys with the Xcel I card, regardless of whether SKS is used.

For Xcel II cards, and when no Xcel card is installed, a key length of 1024 bits or less is recommended. While larger private keys are supported, 2048-bit and larger private keys can have a significant impact on performance.

Note that if you install an Xcel card in an Equalizer that already has HTTPS clusters with certificates defined, you must delete the HTTPS clusters and add them again in order to store the private keys on the Xcel card in SKS.

Caution – If you do not check the **sks** box when uploading a cluster certificate to an Xcel I card, your cluster certificate's private key is kept on Equalizer (in the directory `/var/eq/ssl`) and will be accessible to anyone who can log into Equalizer. This also applies to all certificate keys uploaded when using an Xcel II card, which does not support SKS. It is therefore essential that you restrict the ability of non-authorized personnel to access Equalizer, since any user can log in and copy or remove your private key. All Equalizer logins should be password protected with non-trivial passwords to restrict access to your private keys, and passwords should be given only to trusted personnel.

Clearing Secure Key Storage

Over time, it is possible for the SKS memory on an Xcel I card to become full. When SKS is full, the following error is returned when you try to add another key (or replace an existing key):

```
Call to 'cert2sks' failed.
Error initializing RSA material
Using stdin
Could not allocate RSA key (N8_NO_MORE_RESOURCE).
Died at /usr/local/sbin/cert2sks line 286.
```

When this happens, you can do one of two things:

- Uncheck the **use secure key storage** check box when adding the SSL certificate; the private key will be kept on the Equalizer instead of in SKS.

- Clear SKS memory (using the procedure below); this removes all keys from SKS and will free up any space taken by keys that are no longer used (assuming you have not already used all 128kb of space on the Xcel card with valid keys). After you clear SKS, you'll need to re-add all the certificates for all the HTTPS clusters whose keys were kept in SKS.

To clear SKS memory on the Xcel card:

1. Log into Equalizer as *root* over the serial line, or login via SSH and use the **su** command to switch to the *root* login.
2. Enter the following command:

```
SKSManager -R -u 0
```
3. After the operation completes (which should take about 1 minute), re-add all certificates for all HTTPS clusters.

Configuring Cipher Suites

The **cipher suite** HTTPS cluster parameter lists the supported encryption algorithms for incoming HTTPS requests. If a client request comes into Equalizer that does not use a cipher in this list, the connection is refused. If this field is blank, then any cipher suite supported by Equalizer's SSL implementation or an optionally installed Xcel Card will be accepted.

To view or set the **cipher suite** field for a cluster, click on the cluster name in the left frame and then select the **Security > SSL** tab in the right frame.

Default Cipher Suites

For an Equalizer with no Xcel SSL Accelerator Card installed and for systems with an Xcel II (newer generation) Card installed, the following default setting for **cipher suite** is used:

```
AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA
```

For an Equalizer with an Xcel I (older generation) Card installed, the following default value is used:

```
DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA
```

Updating the Cipher Suites Field

This field can be used to specify a custom cipher suite required by the servers in a cluster. For example, if your servers are required to support medium and high encryption using SSLv3 *only*, you could specify the following string for **cipher suite**, which will cause all non-SSLv3 client requests to be refused:

```
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:!LOW:!SSLv2:+SSLv3:+EXP:+eNULL
```

The **cipher suite** field requires a string in the format of the Apache **mod_ssl** directive **SSLCipherSuite** (for more information and examples, see http://httpd.apache.org/docs/2.0/mod/mod_ssl.html#sslcipherSuite).

The tables in the following sections list the cipher suites supported by Equalizer. Also see the discussion of the cluster parameter "**cipher suite**" on page 75.

No Xcel and Xcel II Card Cipher Suites

The following cipher suites are supported by the base Equalizer software and by the Xcel II (newer generation) SSL Accelerator Card:

OpenSSL Cipher Suite Name	TLS/SSL Cipher Suite Names
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
RC4-SHA	TLS_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_RC4_128_MD5
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
The cipher suites below are supported but are not recommended. (In earlier releases, the EXP-RC4-MD5 ciphers were included by default in cipher suite for older browsers that only support 40-bit encryption. If some clients for your web services support only 40-bit encryption, then add EXP-RC4-MD5 to the cipher suite list.)	
EXP-RC4-MD5	TLS_RSA_EXPORT_WITH_RC4_40_MD5 SSL_RSA_EXPORT_WITH_RC4_40_MD5 SSL_CK_RC4_128_EXPORT40_WITH_MD5

Xcel I Card Cipher Suites

The following cipher suites are supported by the older generation Xcel I SSL Accelerator card.

OpenSSL Cipher Suite Name	TLS/SSL Cipher Suite Names
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
RC4-SHA	TLS_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_SH
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_RC4_128_MD5