Cisco TelePresence VCS Certificate Creation and Use

Deployment Guide

Cisco VCS X8.1

D14548.08

December 2013

Contents

Introduction

This deployment guide provides instructions on how to create X.509 cryptographic certificates for use with the Cisco TelePresence Video Communication Server (VCS), and how to load them into VCS.

PKI introduction

Public Key Infrastructure (PKI) provides the mechanisms through which communications can be secured (encrypted and integrity protected) and identities can be verified. Underlying PKI is:

- **A public/private key pair**: a public key is used to encrypt data sent to a server, but only the private key (kept secret by the server) can be used to decrypt it.
- **Signatures of data**: data can be "signed" by a server, by using a combination of a cryptographic hash of the data and the server's private key. A client can verify the signature by using the server's public key and verifying the same hash. This ensures the data has been sent from the expected server, and has not been tampered with.
- **Certificates**: a certificate is a wrapper around a public key, and provides information about the owner of the key. This metadata is provided in X.509 format, and typically includes the server name and contact details for the owner.
- **A certificate chain**: a certificate can be signed by a Certificate Authority (CA) using its own private key. In turn, therefore, a certificate can be verified as being signed by a CA by checking the signature against the CA's certificate (public key). Web browsers and other clients have a list of CA certificates that they trust, and can thus verify the certificates of individual servers.

Transport Layer Security (TLS) is the standard mechanism for securing a TCP connection between hosts on a TCP/IP network. For example, secure HTTP (HTTPS) uses TLS to encrypt and verify traffic. To establish a TLS connection:

1. An initial TCP connection is made, and the client sends its capabilities (including cipher suites) and a random number.
2. The sever responds with its choice of those capabilities, another random number, and its certificate.
3. The client verifies that the server certificate was issued (signed) by a CA that it trusts, and has not been revoked.
4. The client sends a "pre-master secret", encrypted with the server's public key.
5. This pre-master secret, combined with the exchanged random numbers (to prevent replay attacks), is used to generate a "master secret", with which the remaining communications of this TLS session are encrypted between the client and server.

The following sections describe how these PKI components can be used with the VCS.

Overview of certificate use on the VCS

VCS needs certificates for:

- Secure HTTP with TLS (HTTPS) connectivity
- TLS connectivity for SIP signaling, endpoints and neighbor zones
- Connections to other systems such as Cisco Unified CM, Cisco TMS, LDAP servers and syslog servers

It uses its list of trusted Certificate Authority (CA) certificates and associated certificate revocation lists (CRLs) to validate other devices connecting to it.

It uses the Server Certificate and the Private key to provide a signed certificate to provide evidence that the VCS is the device it says it is. This can be used with neighboring devices such as Microsoft Lync or Cisco Unified CM, as well as administrators using the web interface.

A certificate identifies the VCS. It contains names by which it is known and to which traffic is routed. If the VCS is known by multiple names for these purposes, such as if it is part of a cluster, this must be represented in the X.509 subject data, according to the guidance of RFC5922. The certificate must contain the FQDN of both the VCS itself and of the cluster. If a certificate is shared across cluster peers, it must list all possible peer FQDNs. The following lists show what must be included in the X.509 subject, depending on the deployment model chosen.

If the VCS is not clustered:

- Subject Common Name = FQDN of VCS
- Subject Alternate Names = leave blank

If the VCS is clustered, with individual certificates per VCS:

- Subject Common Name = FQDN of VCS
- Subject Alternate Names = FQDN of VCS, FQDN of cluster

Wildcard certificates manage multiple subdomains and the services names they support, they can be less secure than SAN (Subject Alternate Name) certificates. VCS does not support wildcard certificates.

Certificate generation overview

X.509 certificates may be supplied from a third party, or may be generated by a certificate generator such as OpenSSL. Third-party certificates supplied by recognized certificate authorities are recommended, although VCS deployments in controlled or test environments can use internally generated certificates.

Certificate generation is usually a 3-stage process:

- Stage 1: generate a private key
- Stage 2: create a certificate request
- Stage 3: authorize and create the certificate

This document presents alternative methods of generating the root certificate, client/server certificate for the VCS, and private key:

- Generating a certificate signing request (CSR) [p.5] describes how to use the VCS itself to generate the private key and certificate request.
- Appendix 2: Certificate generation using OpenSSL only [p.12] documents the OpenSSL-only process, which could be used with a third party or internally managed CA.

For mutual TLS authentication the VCS **Server** certificate must be capable of being used as a **Client** certificate as well, thus allowing the VCS to authenticate as a client device to a neighboring server.

Generating a certificate signing request (CSR)

A CSR contains the identity information about the owner of a private key. It can be passed to a third-party or internal certification authority for generating a signed certificate, or it can be used in conjunction with an application such as OpenSSL.

Creating a CSR using VCS

The VCS can generate server certificate signing requests. This removes the need to use an external mechanism to generate and obtain certificate requests.

To generate a CSR:

1. Go to **Maintenance > Security certificates > Server certificate**.
2. Click **Generate CSR** to go to the **Generate CSR** page.
3. Enter the required properties for the certificate.
   - See Server certificates and clustered systems [p.5] if your VCS is part of a cluster.
   - See Server certificates and Unified Communications [p.6] if this VCS is part of a Unified Communications solution.
   - The certificate request includes automatically the public key that will be used in the certificate, and the client and server authentication Enhanced Key Usage (EKU) extension.
4. Click **Generate CSR**. The system will produce a signing request and an associated private key. Note that the private key is stored securely on the VCS and cannot be viewed or downloaded. You must never disclose your private key, not even to the certificate authority.
5. You are returned to the **Server certificate** page. From here you can:
   - **Download** the request to your local file system so that it can be sent to a certificate authority. You are prompted to save the file (the exact wording depends on your browser).
   - View the current request (click **Show (decoded)** to view it in a human-readable form, or click **Show (PEM file)** to view the file in its raw format).

Note that only one signing request can be in progress at any one time. This is because the VCS has to keep track of the private key file associated with the current request. To discard the current request and start a new request, click **Discard CSR**.

You must now authorize the request and generate a signed PEM certificate file. You can pass it to a third-party or internal certification authority, or use it in conjunction with an application such as OpenSSL (see Operating as a Certificate Authority using OpenSSL [p.13]).

When the signed server certificate is received back from the certificate authority, it must be uploaded to the VCS as described in Loading certificates and keys onto VCS [p.7].

Server certificates and clustered systems

When a CSR is generated, a single request and private key combination is generated for that peer only.

If you have a cluster of VCSs, you must generate a separate signing request on each peer. Those requests must then be sent to the certificate authority and the returned server certificates uploaded to each relevant peer.

You must ensure that the correct server certificate is uploaded to the appropriate peer, otherwise the stored private key on each peer will not correspond to the uploaded certificate.

Server certificates and Unified Communications

As a prerequisite to requesting server certificates for use in Unified Communications deployments, you must ensure that mobile and remote access is enabled on all VCSs, and that they are configured with the relevant domains. See *Unified Communications: Mobile and Remote Access via VCS Deployment Guide*.

The following sections summarize the server certificate requirements for VCS Control and VCS Expressway.

VCS Control server certificate requirements

The VCS Control server certificate may need to include the following elements in its list of subject alternate names:

- The **Chat Node Aliases** that are configured on the IM and Presence servers. They will be required only for Unified Communications XMPP federation deployments that intend to use both TLS and group chat. (Note that Unified Communications XMPP federation will be supported in a future VCS release).
  The VCS Control automatically includes the chat node aliases in the CSR, providing it has discovered a set of IM&P servers.
- The names, in FQDN format, of all of the **Phone Security Profiles** in Cisco Unified CM that are configured for encrypted TLS and are used for devices requiring remote access. This ensures that Cisco Unified CM can communicate with VCS Control via a TLS connection when it is forwarding messages from devices that are configured with those security profiles.

A new certificate may need to be produced if chat node aliases are added or renamed, such as when an IM and Presence node is added or renamed, or if new TLS phone security profiles are added. You must restart the VCS Control for any new uploaded server certificate to take effect.

VCS Expressway server certificate requirements

The VCS Expressway server certificate may need to include the following elements in its list of subject alternate names:

- All of the domains which have been configured for Unified Communications. They are required for secure communications between endpoint devices and VCS Expressway.
  They may include the email address domain entered by users of the client application (e.g. Jabber) and any presence domains (as configured on the VCS Control) if they are different. There is no need to include the domains in DNS-SEC deployments.
- The same set of **Chat Node Aliases** as entered on the VCS Control's certificate, if you are deploying federated XMPP.
  Note that the list of required aliases can be viewed (and copy-pasted) from the equivalent **Generate CSR** page on the VCS Control.

A new certificate must be produced if new presence domains or chat node aliases are added to the system. You must restart the VCS Expressway for any new uploaded server certificate to take effect.

Loading certificates and keys onto VCS

The VCS uses standard X.509 certificates. The certificate information must be supplied to the VCS in PEM format. Typically 3 elements are loaded:

- The server certificate (which is generated by the certificate authority, identifying the ID of the certificate holder, and should be able to act as both a client and server certificate).
- The private key (used to sign data sent to the client, and decrypt data sent from the client, encrypted with the public key in the server certificate). This must only be kept on the VCS and backed up in a safe place – security of the TLS communications relies upon this being kept secret.
- A list of certificates of trusted certificate authorities.

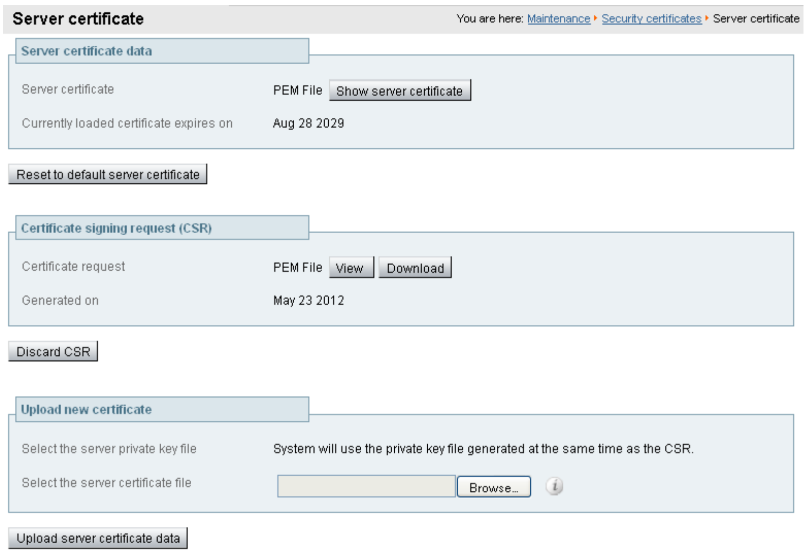
Note: New installations of VCS software (from X8.1 onwards) ship with a temporary trusted CA, and a server certificate issued by that temporary CA. We strongly recommend that you replace the server certificate with one generated by a trusted certificate authority, and that you install CA certificates for the authorities that you trust.

Loading a server certificate and private key onto VCS

The VCS's server certificate is used to identify the VCS when it communicates with client systems using TLS encryption, and with web browsers over HTTPS.

To upload a server certificate:

1. Go to **Maintenance > Security certificates > Server certificate**.
2. Use the **Browse** button to select and upload the **server certificate** PEM file.
3. If you used an external system to generate the certificate request you must also upload the **server private key** PEM file that was used to encrypt the server certificate. (The private key file will have been automatically generated and stored earlier if the VCS was used to produce the signing request for this server certificate.)
   - The **server private key** must not be password protected.
   - You cannot upload a server private key if a certificate signing request is in progress.
4. Click **Upload server certificate data**.


Server certificate
You are here: Maintenance ▸ Security certificates ▸ Server certificate

Server certificate data

Server certificate	PEM File Show server certificate
Currently loaded certificate expires on	Aug 28 2029

Reset to default server certificate

Certificate signing request (CSR)

Certificate request	PEM File View Download
Generated on	May 23 2012

Discard CSR

Upload new certificate

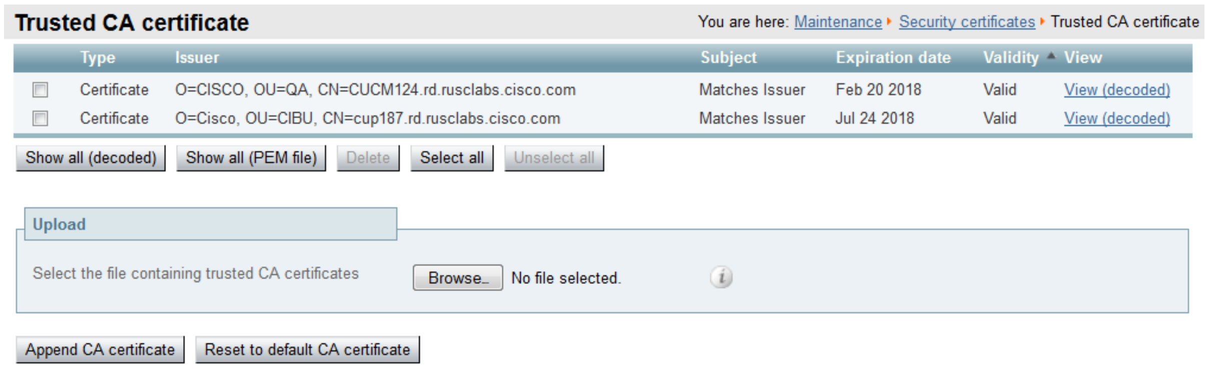
Select the server private key file	System will use the private key file generated at the same time as the CSR.
Select the server certificate file	Browse...

Upload server certificate data

Managing the trusted CA certificate list

The **Trusted CA certificate** page (**Maintenance > Security certificates > Trusted CA certificate**) allows you to manage the list of certificates for the Certificate Authorities (CAs) trusted by this VCS. Certificates presented to the VCS must be signed by a trusted CA on this list and there must be a full chain of trust (intermediate CAs) to the root CA.

- To upload a new file of CA certificates, **Browse** to the required PEM file and click **Append CA certificate**. This will append any new certificates to the existing list of CA certificates. Note that if you are replacing existing certificates for a particular issuer and subject, you have to manually delete the previous certificates.
- To replace all of the currently uploaded CA certificates with the system's original list of trusted CA certificates, click **Reset to default CA certificate**.
- To view the entire list of currently uploaded trusted CA certificates, click **Show all (decoded)** to view it in a human-readable form, or click **Show all (PEM file)** to view the file in its raw format.
- To view an individual trusted CA certificate, click on **View (decoded)** in the row for the specific CA certificate.
- To delete one or more CA certificates, tick the box(es) next to the relevant CA certificate(s) and click **Delete**.


Trusted CA certificate
You are here: Maintenance ▸ Security certificates ▸ Trusted CA certificate

	Type	Issuer	Subject	Expiration date	Validity	View
☐	Certificate	O=CISCO, OU=QA, CN=CUCM124.rd.rusclabs.cisco.com	Matches Issuer	Feb 20 2018	Valid	View (decoded)
☐	Certificate	O=Cisco, OU=CIBU, CN=cup187.rd.rusclabs.cisco.com	Matches Issuer	Jul 24 2018	Valid	View (decoded)

Show all (decoded)
Show all (PEM file)
Delete
Select all
Unselect all
Upload
Select the file containing trusted CA certificates
Browse...
No file selected.
Append CA certificate
Reset to default CA certificate

Managing certificate revocation lists (CRLs)

Certificate revocation list (CRL) files are used by the VCS to validate certificates presented by client browsers and external systems that communicate with the VCS over TLS/HTTPS. A CRL identifies those certificates that have been revoked and can no longer be used to communicate with the VCS.

We recommend that you upload CRL data for the CAs that sign TLS/HTTPS client and server certificates. When enabled, CRL checking is applied for every CA in the chain of trust.

CRL sources

The VCS can obtain CRL information from multiple sources:

- automatic downloads of CRL data from CRL distribution points
- through OCSP (Online Certificate Status Protocol) responder URIs in the certificate to be checked (SIP TLS only)
- manual upload of CRL data
- CRL data embedded within the VCS's **Trusted CA certificate** file

The following limitations and usage guidelines apply:

- when establishing SIP TLS connections, the CRL data sources are subject to the **Certificate revocation checking** settings on the **SIP** configuration page
- automatically uploaded CRL files override any manually loaded CRL files (except for when verifying SIP TLS connections, when both manually uploaded or automatically downloaded CRL data may be used)
- when validating certificates presented by external policy servers, the VCS uses manually loaded CRLs only
- when validating TLS connections with an LDAP server for remote login account authentication, the VCS uses CRL data within the **Trusted CA certificate** only

Automatic CRL updates

We recommend that the VCS is configured to perform automatic CRL updates. This ensures that the latest CRLs are available for certificate validation.

To configure the VCS to use automatic CRL updates:

1. Go to **Maintenance > Security certificates > CRL management**.
2. Set **Automatic CRL updates** to *Enabled*.
3. Enter the set of **HTTP(S) distribution points** from where the VCS can obtain CRL files. Note that:
   - you must specify each distribution point on a new line
   - only HTTP(S) distribution points are supported; if HTTPS is used, the distribution point server itself must have a valid certificate
   - PEM and DER encoded CRL files are supported
   - the distribution point may point directly to a CRL file or to ZIP and GZIP archives containing CRL files
4. Enter the **Daily update time** (in UTC). This is the approximate time of day when the VCS will attempt to update its CRLs from the distribution points.
5. Click **Save**.

Manual CRL updates

CRL files can also be uploaded manually to the VCS. Certificates presented by external policy servers can only be validated against manually loaded CRLs.

To upload a CRL file:

1. Go to **Maintenance > Security certificates > CRL management**.
2. Click **Browse** and select the required file from your file system. It must be in PEM encoded format.
3. Click **Upload CRL file**.
   This uploads the selected file and replaces any previously uploaded CRL file.

Click **Remove revocation list** if you want to remove the manually uploaded file from the VCS.

Note that if a certificate authority's CRL expires, all certificates issued by that CA will be treated as revoked.

Configuring revocation checking for SIP TLS connections

You must also configure how certificate revocation checking is managed for SIP TLS connections.

1. Go to **Configuration > SIP**.
2. Scroll down to the **Certificate revocation checking** section and configure the settings accordingly:

Field	Description	Usage tips
Certificate revocation checking mode	Controls whether revocation checking is performed for certificates exchanged during SIP TLS connection establishment.	We recommend that revocation checking is enabled.
Use OCSP	Controls whether the Online Certificate Status Protocol (OCSP) may be used to perform certificate revocation checking.	To use OCSP, the X.509 certificate to be checked must contain an OCSP responder URI.
Use CRLs	Controls whether Certificate Revocation Lists (CRLs) are used to perform certificate revocation checking.	CRLs can be used if the certificate does not support OCSP.
Allow CRL downloads from CDPs	Controls whether the download of CRLs from the CDP URIs contained in X.509 certificates is allowed.	
Fallback behavior	Controls the revocation checking behavior if the revocation status cannot be established, for example if the revocation source cannot be contacted. *Treat as revoked*: treat the certificate as revoked (and thus do not allow the TLS connection). *Treat as not revoked*: treat the certificate as not revoked. Default: *Treat as not revoked*	*Treat as not revoked* ensures that your system continues to operate in a normal manner if the revocation source cannot be contacted, however it does potentially mean that revoked certificates will be accepted.

Appendix 1: Troubleshooting

SIP TLS negotiation failures on neighbor and traversal zones

If **TLS verify mode** is enabled, the neighbor system's FQDN or IP address, as specified in the **Peer address** field of the zone's configuration, is used to verify against the certificate holder's name contained within the X.509 certificate presented by that system. (The name has to be contained in either the Subject Common Name or the Subject Alternative Name attributes of the certificate.) The certificate itself must also be valid and signed by a trusted certificate authority.

Therefore when certificates have been generated with peer or cluster FQDNs, ensure that the zone's **Peer address** fields are configured with FQDNs rather than IP addresses.

Certificates with key length of 8192 bits

SIP TLS zones may fail to become active if certificates with a key length of 8192 bits are used. We recommend using certificates with a key length of 4096 bits.

Service failures when using mobile and remote access

Unified Communications mobile and remote access services can fail due to certificate errors if you have uploaded a private key file that does not contain a trailing newline character.

Ensure that the private key file contains a trailing newline character.

Appendix 2: Certificate generation using OpenSSL only

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

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

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

OpenSSL and Mac OS X or Linux

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

OpenSSL and Windows

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

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

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

Creating a certificate request using OpenSSL

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

From a command prompt:

1. For Windows: change to the directory where OpenSSL is installed (typically a ‘bin’ directory).
   For Mac OS X: stay in the root of the user’s directory.
2. Personalize the openssl.cfg file:
   a. For Windows: copy **openssl.cfg** to **openssl_request.cfg**
      For Mac OS X: copy **/System/Library/OpenSSL/openssl.cnf** to the root of the user’s directory as **openssl_request.cfg**
   b. Use a text editor to edit the openssl_request.cfg file that was created by the above copy command, and ensure that the line “**`req_extensions = v3_req # The extensions to add to a certificate request`**” does not have a # at the beginning of the line. Delete the # if it is there.
   c. Scroll down to the “**`[ v3_req ]`**” section and below this section title add:
      `extendedKeyUsage=serverAuth, clientAuth`
   d. Save the file.

3. If the certificate is for a cluster of VCSs:
   a. Under the same "`[ v3_req ]`" section add:
      `subjectAltName="DNS:<FQDN of VCS cluster>,DNS:<FQDN of peer 1>,DNS:<FQDN of peer 2>,DNS:<FQDN of peer n>"`
      as the bottom line of this section (before "[v3_ca]"), filling in the details for the VCS deployment as appropriate (cluster FQDN and FQDNs of all peers).
   b. Save the file.
4. Generate a private key by running the following command:
   `openssl genrsa -out privatekey.pem 2048`
   The **privatekey.pem** file will be used to create the certificate request and will also be required for loading into the VCS. The file is created in the directory that the openssl command is run from.
5. Generate a certificate request (suitable for use with a Certification Authority) by running the following command:
   `openssl req -new -key privatekey.pem -config openssl_request.cfg -out certcsr.der -outform DER -sha1`
6. Enter the data requested, including:
   - Country
   - State or province
   - Locality name
   - Organization name
   - Organizational unit
   - Common name - this is the VCS cluster FQDN if the certificate is for a cluster of VCSs or it is the FQDN of the VCS if the certificate is for a single VCS
   - Email address - optional, can leave blank
   - A challenge password - optional, can leave blank
   - An optional company name - optional, can leave blank

   After entering the requested data, the certificate request file **certcsr.der** is now available.

To validate that DNS entries have been entered correctly into the request, the **certcsr.der** file can be decoded using the command:

```
openssl req -text -noout -in certcsr.der -inform DER
```

This certificate request file can be passed to an internal or third-party Certificate Authority for generating the X.509 certificate. OpenSSL can be used to operate a private CA, as described below.

Operating as a Certificate Authority using OpenSSL

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

If you have already configured OpenSSL to act as a CA, go to section Creating a signed certificate using OpenSSL [p.15].

Configuring OpenSSL to act as a CA

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

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

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

For example, use the commands:

```
mkdir demoCA
cd demoCA
mkdir certs
mkdir newcerts
mkdir private
touch index.txt
echo 10 > serial
```

Creating a Certificate Authority using OpenSSL

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

From a command prompt:

1. Ensure that you are in the **demoCA** directory.
2. For Windows: copy **openssl.cfg** from the directory where OpenSSL is installed to the **demoCA** directory and rename it as **openssl_local.cfg**.
   For Mac OS X: copy **/System/Library/OpenSSL/openssl.cnf** to the **demoCA** directory and rename it as **openssl_local.cfg**.
3. Use a text editor to edit the **openssl_local.cfg** file that was created by the above copy command. Make the following modifications to the **`[CA_default]`** section:
   a. Ensure that the line **`copy_extensions = copy`** does not have a # at the beginning of the line. Delete the # if it is there. If the line remains commented out, it will strip attributes in the CSR and the SSL Server and SSL Client attributes will not appear in the certificate.
   b. Change **`policy = policy_match`** to **`policy = policy_anything`**
   c. Change **`dir = ./demoCA`** to **`dir = .`**
   d. Optionally, change **`default_days = 365`** (1 year validity of the generated certificate) to **`default_days = 3650`** (10 years, or choose another suitable value).
   e. Save the file.
4. Generate a private key for the CA by running the following command:
   `openssl genrsa -aes256 -out private/cakey.pem 4096`
   This will prompt for a password with which to encrypt the private key: choose a strong password and record it in a safe place. The cakey.pem file will be used to create the CA certificate and to sign other certificates and must also be kept secure.
5. Generate the CA certificate by running the following command.
   For Windows: **`openssl req -new -x509 -days 3650 -key private/cakey.pem -config openssl_local.cfg -sha1 -extensions v3_ca -out cacert.pem`**
   For OS X: **`openssl req -new -x509 -days 3650 -key private/cakey.pem -config openssl_local.cfg -sha1 -extensions v3_ca -out cacert.pem`**
6. Enter a passphrase for the key, and then enter the data requested, including:
   - Country
   - State or province
   - Locality name

- Organization name
- Organizational unit
- Common name – this is typically the name of a contact person for this CA
- Email address – optional, can leave blank

After entering the requested data, the operation completes and the certificate authority certificate **cacert.pem** is now available.

Creating a signed certificate using OpenSSL

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

From a command prompt:

1. Ensure that you are in the **demoCA** directory.
2. Ensure that the certificate request file (**certcsr.pem**) is available:
   - If the certificate request was created using the VCS (recommended process):
     Copy the file downloaded from the VCS into the **demoCA** directory and rename it as **certcsr.pem**.
   - If the certificate request was created using OpenSSL:
     Copy the previously generated certificate request into the **demoCA** directory and then covert it to PEM format by running the following command:
     ```
     openssl req -in certcsr.der -inform DER -out certcsr.pem -outform PEM
     ```
3. Generate a signed server certificate by running the following command:
   ```
   openssl ca -config openssl_local.cfg -cert cacert.pem -keyfile
   private/cakey.pem -in certcsr.pem -out certs/server.pem -md sha1
   ```
   If you receive a "failed to update database TXT_DB error number 2" error message, you can remove the contents of the **index.txt** file and then rerun the command.
4. You will be prompted to enter the password for the CA's private key.

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

Creating self-signed certificates using OpenSSL

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

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

Appendix 3: Converting a DER certificate file to PEM format

A private key, root (CA) certificate and the server / client certificate can be generated using third-party tools (or purchased from a certificate authority), and may be generated as PEM (required format, extension .pem) or DER (extension .cer) format files.

Certificates must be in PEM format for use on the VCS. Conversion from DER to PEM format can be done in one of two ways, either using OpenSSL or Windows, as documented in the following sections.

Converting a DER certificate file to a PEM file using OpenSSL

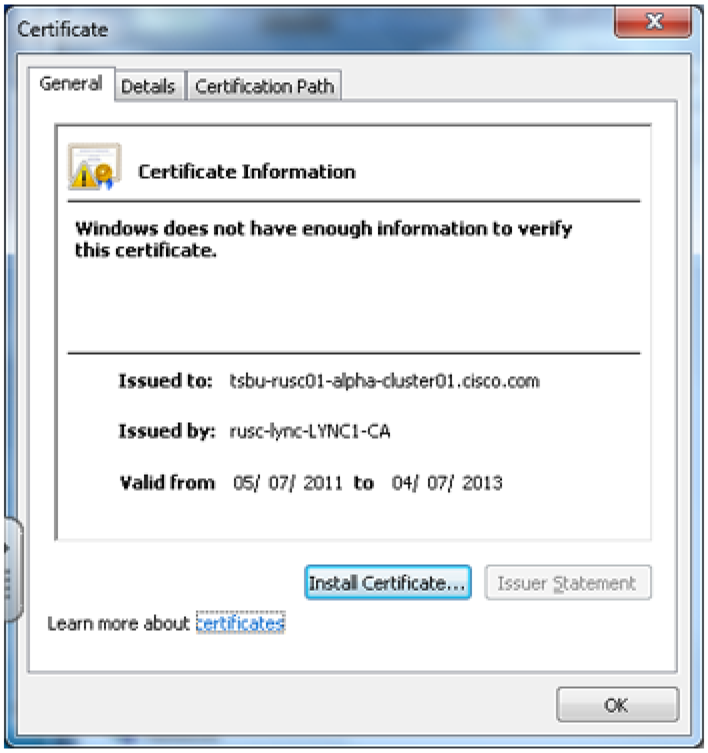
To convert from DER to PEM format, on a system running openssl, execute the command:

```
openssl x509 -in <filename>.cer -inform DER -out <filename>.pem -outform PEM
```

Converting a DER certificate file to a PEM file using Microsoft Windows

To convert from DER to PEM format using Microsoft Windows:

1. Double click on the DER file to convert (this will likely have a '.cer' extension).



Certificate

Certificate Information

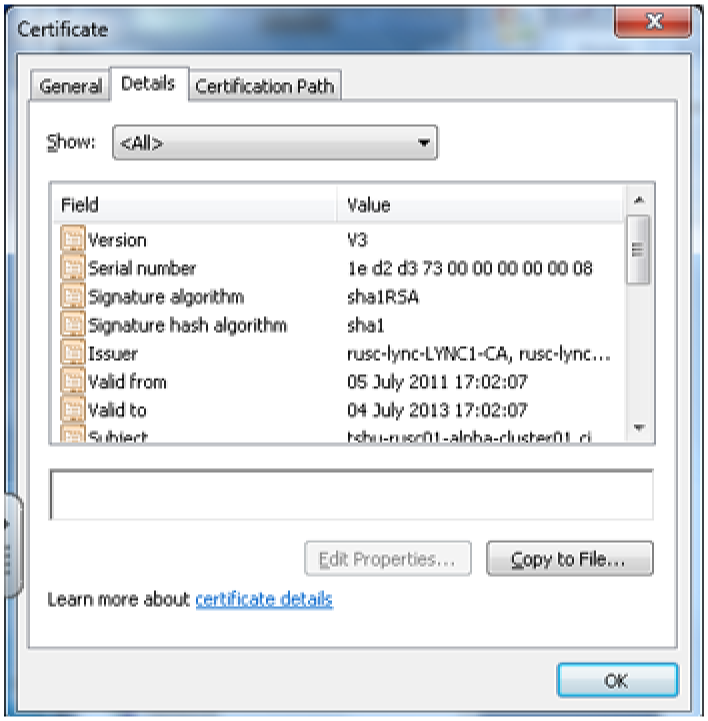
Windows does not have enough information to verify this certificate.

Issued to: tsbu-rusc01-alpha-cluster01.cisco.com

Issued by: rusc-lync-LYNC1-CA

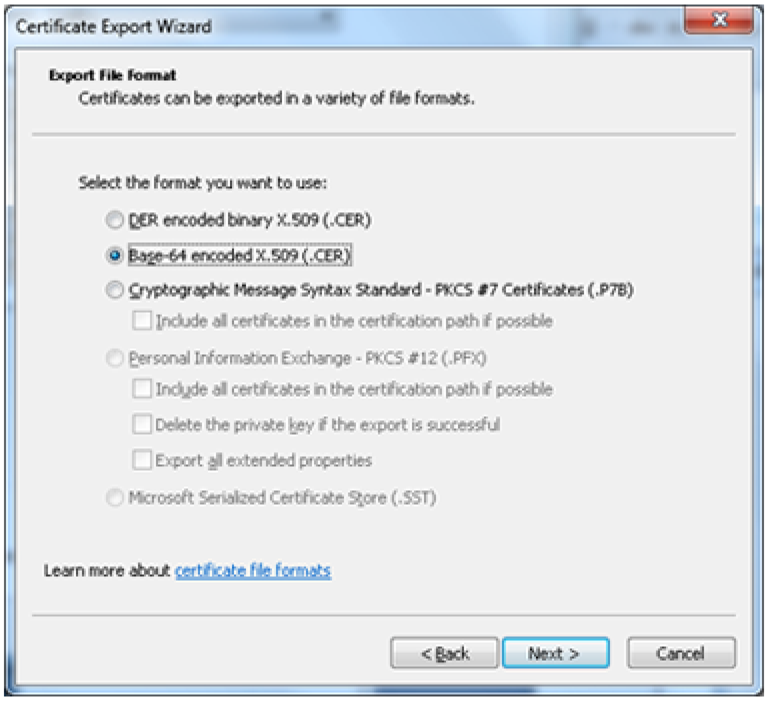
Valid from 05/ 07/ 2011 to 04/ 07/ 2013

2. Select the **Details** tab.



Field	Value
Version	V3
Serial number	1e d2 d3 73 00 00 00 00 00 08
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	rusc-lync-LYNC1-CA, rusc-lync...
Valid from	05 July 2011 17:02:07
Valid to	04 July 2013 17:02:07
Subject	tshu-rusc01-alpha-cluster01.ci

3. Click **Copy to File…**
4. On the **Welcome** page, click **Next**.
5. Select *Base-64 encoded X.509 (.CER)* and click **Next**.


Certificate Export Wizard
< Back
Next >
Cancel
Learn more about certificate file formats

Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:

- DER encoded binary X.509 (.CER)
- [✓ Base-64 encoded X.509 (.CER)]
- Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - Include all certificates in the certification path if possible
- Personal Information Exchange - PKCS #12 (.PFX)
  - Include all certificates in the certification path if possible
  - Delete the private key if the export is successful
  - Export all extended properties
- Microsoft Serialized Certificate Store (.SST)

6. Click **Browse** and select required destination for file (e.g. **server.pem**) and then click **Next**.
7. Click **Finish**.
8. Change the filename from **server.pem.cer** to **server.pem**.
9. This will be used in the Loading certificates and keys onto VCS [p.7] section of this document.

Appendix 4: Decoding certificates

This section describes some methods for decoding and viewing the content of certificates.

OpenSSL

A PEM file (e.g. **cert.pem**) can be decoded by the following command:

```
openssl x509 -text -in cert.pem
```

A DER file (e.g. **cert.cer**) can be decoded by the following command:

```
openssl x509 -text -inform DER -in cert.cer
```

Firefox

The certificate in use for a website being visited can be viewed in Firefox by clicking on the security information button on the address bar, and then clicking **More Information** followed by **View Certificate**.

Internet Explorer

The certificate in use for a website being visited can be viewed in Internet Explorer by clicking the lock icon to the right of the address bar. A **Website Identification** dialog will appear. Click the **View Certificates** link at the bottom.

Document revision history

The following table summarizes the changes that have been applied to this document.

Revision	Date	Description
8	December 2013	Updated for X8.1. Removed Microsoft Lync-based certificate generation guidelines (they can now be found in *Microsoft Lync and VCS Deployment Guide*. Various improvements and clarifications to "Certificate generation using OpenSSL only" appendix.
7	February 2013	Added sections on CRL management, troubleshooting, and how to configure Windows Server Manager with a "client and server" certificate template.
6	August 2012	Updated for VCS X7.2 functionality to generate certificate signing requests.
5	February 2012	Major clarifications and updates, including OpenSSL-specific section.
4	December 2011	Minor updates for clarification.
3	September 2011	Updated for Microsoft Lync 2010 (Lync).
2	October 2010	New document styles applied. New appendices added for decoding certificates and guidance on generating certificates for use with Microsoft Office Communications Server (OCS).
1	November 2009	Initial release.