Cisco Meeting Server

Cisco Meeting Server Release 2.4 and later
Certificate Guidelines for Single Split Server Deployments

February 05, 2019
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 5, 2019</td>
<td>Correction to pki csr example command.</td>
</tr>
<tr>
<td>January 21, 2019</td>
<td>Invalid example csr name corrected.</td>
</tr>
<tr>
<td>January 02, 2019</td>
<td>Changed title to “…2.4 and later”, no changes for version 2.5.</td>
</tr>
<tr>
<td>December 14, 2018</td>
<td>Minor correction.</td>
</tr>
<tr>
<td>November 29, 2018</td>
<td>Minor correction.</td>
</tr>
<tr>
<td>October 02, 2018</td>
<td>Released for Cisco Meeting Server 2.4. Announced removal of H.323 Gateway, SIP Edge, TURN Server, XMPP Server and Load Balancer components in future version of Meeting Server software.</td>
</tr>
<tr>
<td>January 12, 2018</td>
<td>Released for Cisco Meeting Server 2.3</td>
</tr>
<tr>
<td>December 19, 2016</td>
<td>Minor corrections and released for 2.1</td>
</tr>
<tr>
<td>August, 03, 2016</td>
<td>Rebranded for Cisco Meeting Server 2.0.</td>
</tr>
</tbody>
</table>
1 Introduction

The Cisco Meeting Server software can be hosted on specific servers based on Cisco Unified Computing Server (UCS) technology as well as on the Acano X-Series hardware, or on a specification-based VM server. Cisco Meeting Server is referred to as the Meeting Server throughout this document.

The Cisco Meeting Server is very secure, most of the services and applications running on the server use the TLS cryptographic protocol for communication. TLS allows communicating parties to exchange X.509 certificates and public keys in order authenticate the other party, and exchange encryption algorithms to encrypt data transmitted between the parties.

This Certificate Guidelines document explains how to create and install certificates for a Single Split Server deployment.

Note: The Cisco Meeting Server software is referred to as the Meeting Server throughout the remainder of this guide.

1.1 How to use this guide

The remainder of this chapter explains concepts that you will need to understand in order to deploy certificates across the Meeting Server deployment. Skip this if you are already familiar with PKI, certificates, and trust stores.

Chapter 1 details where certificates are required within the single split server model, and the type of certificates required.

Chapter 3 explains how to create certificates.

Chapter 4 covers installing the certificates on the Meeting Server.

Chapter 5 provides troubleshooting information for typical certificate related issues.

Chapter 6 explains how you can quickly create self-signed certificates.

Appendix A covers using OpenSSL rather than the Meeting Server pki command if you prefer to use OpenSSL.

Appendix B provides an overview of permitted filename extensions for certificate files and private keys.

Appendix C lists the MMP pki commands.

This guide is part of the documentation set (shown in Figure 1) for the Meeting Server.
1 Introduction

Figure 1: Overview of guides covering the Cisco Meeting Server

These documents can be found on cisco.com.
1.2 Brief overview of PKI

Public key infrastructure (PKI) provides a mechanism to secure communications and validate identities of communicating parties. Communications are made secure through encryption, and identities are validated through the use of public/private key pairs and digital identity certificates.

1.2.1 Public/private key pair

A public and private key pair comprises two uniquely related cryptographic keys mathematically related. Whatever is encrypted with a public key may only be decrypted by its corresponding private key (which must be kept secret), and vice versa.

1.2.2 Certificates

A certificate is a wrapper around the public key, and provides information about the owner of the public key. It typically contains the name of the entity to which the certificate is issued, contact details for the owner, validity dates (when the certificate is valid), and issuer (the authority that issued the certificate). Certificates need to be signed by trustworthy authorities that can validate that the owner is who they claim to be. Certificate Authorities (CAs) are trustworthy authorities that certify the identities of individuals, organizations, and computers on the network.

When an entity requires a certificate, it first generates a public/private key pair. It then creates a Certificate Signing Request (.csr) file which contains the entity’s public key and information identifying the entity (see Table 1). The entity signs the .csr file using their private key and sends the .csr file to a CA for processing. Depending on the level of verification required, the entity may send the .csr file to either a public CA, such as Verisign, or use an internal CA, for example Active Directory server with the Active Directory Certificate Services Role installed.

The CA uses the .csr file and public key to verify the identity of the entity. If verification is successful, the CA issues a digital identity certificate to the entity, which is proof that the entity named in the certificate is the owner of the public key and private key set. The digital identity certificate is used by the entity to give other entities on the network a high level of assurance that the public key really belongs to the owner of the private key.

Table 1: Information in a .csr file

<table>
<thead>
<tr>
<th>Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name (CN)</td>
<td>This is the fully qualified domain name that you wish to secure e.g. 'www.example.com'.</td>
</tr>
<tr>
<td>Organization or Business name (O)</td>
<td>Usually the legal incorporated name of a company. It should include any suffixes such as Ltd., Inc., or Corp.</td>
</tr>
<tr>
<td>Organizational unit or Department name (OU)</td>
<td>For example, Support, IT, Engineering, Finance.</td>
</tr>
<tr>
<td>Information</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Location (L)</td>
<td>City or town. For example, London, Boston, Milan, Berlin.</td>
</tr>
<tr>
<td>Province, Region, County or State (ST)</td>
<td>For example, Buckinghamshire, New Jersey. Do not abbreviate.</td>
</tr>
<tr>
<td>Country (C)</td>
<td>The two-letter ISO code for the country where your organization is located. For example, US, GB, FR.</td>
</tr>
<tr>
<td>An email address</td>
<td>An email address to contact the organization. Usually the email address of the certificate administrator or IT department.</td>
</tr>
<tr>
<td>Subject Alternative Name (subjectAltName)</td>
<td>From X509 Version 3 (RFC 2459), SSL certificates are allowed to specify multiple names that the certificate should match. subjectAltName (SAN) can contain, for example, email addresses, IP addresses, regular DNS host names.</td>
</tr>
</tbody>
</table>

### 1.2.3 Chain of Trust

When an entity is challenged by another to provide its certificate for authentication, that entity needs to present its own certificate, along with a series of other certificates that establish a link to a Certificate Authority that the challenging party trusts (usually known as the Root Certificate Authority). This hierarchy of certificates, linking an entity’s certificate to a Root CA, is called a ‘chain of trust’. It is quite often the case that a Root CA has signed a certificate for another Certificate Authority (known as an Intermediate CA), which in turn has signed the entity’s certificate. In that case, the entity needs to present both its own certificate and the certificate for this Intermediate CA that has been issued by the Root CA. If the entity only presented its own certificate, without establishing a link to a trusted Root CA, the challenging party will not trust the certificate presented. The series of certificates that link an entity’s certificate to a root CA is known as ‘intermediate certificates’ as they are issued to Intermediate CAs.
To enable connecting devices to verify a chain of trust, every certificate includes two fields: "Issued To" and "Issued By". An intermediate CA will show different information in these two fields, to show a connecting device where to continue checking, if necessary, in order to establish trust. Root CA certificates are "Issued To" and "Issued By" themselves, so no further checking is possible.

For example, if Entity A (web server www.example.com) is challenged for authentication by Entity B (web client), Entity A will need to present its certificate and certificate chain to Entity B.

### Figure 3: Certificate chain for Entity A

| Certificate 1 - Issued To: example.com; Issued By: Intermediate CA 1 |
| Certificate 2 - Issued To: Intermediate CA 1; Issued By: Intermediate CA 2 |
| Certificate 3 - Issued To: Intermediate CA 2; Issued By: Root CA |

Providing Entity B has in its trust store the certificate for Root CA, a secure connection can be established between Entity A and Entity B. Entity B can use Entity A’s public key to encrypt messages and send them to Entity A. Only Entity A has access to the private key, so only Entity A can decrypt the messages.
Note: This process is called "certificate chaining" and intermediate CA certificates are sometimes called "chained certificates".

1.2.4 Certificate bundles

A certificate bundle is a single file (with an extension of .pem, .cer or .crt) holding a copy of the Root CA’s certificate and all intermediate certificates in the chain. The certificates need to be in sequence with the certificate of the Root CA being last in the certificate bundle. External clients (for example web browsers and XMPP clients) require the certificate and certificate bundle to be presented by the Web Bridge and XMPP server respectively, when setting up a secure connection. If Call Bridge establishes a TLS trunk to a SIP peer, then Call Bridge will need to presents its certificate and certificate bundle to the SIP endpoint.

You can create a certificate bundle by using a plain text editor such as notepad. All of the characters including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- tags need to be inserted into the document. There should be no space between the certificates, for example no spaces or extra lines between -----END CERTIFICATE----- of certificate 1 and -----BEGIN CERTIFICATE----- of certificate 2. At the end of the file there should be 1 extra line. Save the file with an extension of .pem, .cer, or .crt.

1.2.5 Trust stores

Web browsers and other clients hold a list of signing authorities that they trust and therefore, by a “chain of trust”, the servers they can trust. These trusted CAs are held in a ‘trust store’ on the client. When the trusted CA issues a revocation list, the client updates its trust store removing the entities in the revocation list from the store.

For a connecting client (or device) to trust a certificate, the client will check whether the CA of the certificate is held in the client’s trust store. If the certificate was not issued by a trusted CA, the connecting client will then check to see if the certificate of the issuing CA was issued by a trusted CA, this will be repeated up the chain until either a trusted CA is found or no trusted CA can be found. If a trusted CA is found, a secure connection will be established between the client and the server. If a trusted CA cannot be found, then the connecting client will usually display an error message.
2 Certificates required for the deployment

This chapter explains where certificates are required to establish secure connections in the single split server deployment, and the type of certificates required.

Note about removing components from Cisco Meeting Server software: The following components will be removed from a future version of the software: H.323 Gateway, SIP Edge, TURN Server, XMPP Server and Load Balancer. You are advised to migrate your deployment over to using the Cisco Expressway as your edge device and to start using the WebRTC app rather than the Cisco Meeting App thick client.

2.1 Using Cisco Expressway as the edge for the Meeting Server

Over the previous few software releases for Cisco Expressway and Meeting Server, edge features have been developed to enable the Expressway to be used as the edge device in Meeting Server deployments. From version 2.4, you should start migrating your Meeting Server deployments from using the Meeting Server SIP edge component (SIP and Lync Call Traversal feature) and the Meeting Server TURN server, to using the Expressway X8.11 TURN server. For details on configuring the Cisco Expressway as the Meeting Server edge device see Cisco Expressway Traffic Classification Deployment guide.

As the edge device for the Meeting Server, the Expressway provides:

- edge support for Microsoft clients on Lync or Skype for Business infrastructure in other organizations, or Skype for Business clients on Office 365 (not "consumer" versions of Skype) connecting to Meeting Server dual homed conferences,

- support for standards based SIP endpoints connecting to Meeting Server hosted conferences,

- edge support for off-premise Cisco Meeting WebRTC App (thin client) using TCP port 443.

Note: The Expressway does not support web proxy of XMPP traffic. Remote Cisco Meeting App thick clients (Windows/Mac desktop or iOS) need to connect to the Load Balancer component of the Meeting Server for XMPP traffic.

Note: In deployments involving on-premise Microsoft infrastructure and the Meeting Server, the Meeting Server must use the Microsoft Edge server to traverse Microsoft calls into and out of the organization.
2.1.1 Expressway Reverse Web Proxy

From X8.10, Expressway includes a Reverse Web Proxy to connect the Cisco Meeting WebRTC App to the Web Bridge of the Meeting Server. You can use this reverse web to avoid the Web Bridge directly connecting to the internet. You need to enable the Web Proxy through the Mobile and Remote Access mode on the Expressway-C and the Expressway-E, but you do not need to completely configure MRA. For information on using the Web Proxy, see the “Cisco Expressway Web Proxy for Cisco Meeting Server Deployment Guide” available under the X8.10 listing here.

Connecting the WebRTC App also needs TURN media relays. Currently, you can use either the TURN server on Expressway-E or the TURN server component on the Meeting Server. But note that the TURN server component on the Meeting Server will be withdrawn in version 3.0 of the Cisco Meeting Server software.

Note: Expressway cannot proxy the XMPP signaling required for the Cisco Meeting App thick clients (desktop and iOS).

2.2 Public or Internal CA signed certificates

Applications on the Meeting Server that interface to external devices, need to be trusted by the external devices, and require certificates signed by a public CA. Applications that interface internally within the Meeting Server only require certificates signed by an internal CA. Internal CA signed certificates can be generated by a local or organizational Certificate Authority, such as an Active Directory server with the Active Directory Certificate Services Role installed, see Section 3.3.

The applications that require public CA signed certificates are shown in Table 2. Applications that only require internal CA signed certificates are shown in Table 3.

For information on using wildcard certificates on the Meeting Server and other certificate related FAQs, go to this link.

Note: In deployments where the Meeting Server is trunked to Cisco Unified Communications Manager, Cisco Unified Communications Manager will accept the Call Bridge certificate signed by a public or internal Certificate Authority; but Cisco Unified Communications Manager requires the Call Bridge certificate to be signed using a template that allows for an Extended Key Usage containing both TLS Web Client Authentication and TLS Web Server Authentication.
Table 2: Public CA signed certificates (single split server model)

<table>
<thead>
<tr>
<th>Applications requiring public CA signed certificate</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Bridge (only if WebRTC Apps are used)</td>
<td>The Cisco Meeting WebRTC App requires a public CA signed certificate from the Web Bridge in order to trust the connection. Note: If using the Expressway Reverse Web Proxy to connect WebRTC apps to the Web Bridge, the Expressway-E certificate must have an entry in the SAN field which matches the hostname (URI) of the Web Bridge. See Section 2.1.1.</td>
</tr>
<tr>
<td>XMPP server (only if native Cisco Meeting Apps are used)</td>
<td>Native Cisco Meeting App require a public CA signed certificate from the XMPP server in order to trust the connection.</td>
</tr>
<tr>
<td>Call Bridge (only if Meeting Server connected on a public network for direct Lync federation)</td>
<td>Lync Edge server requires a public CA signed certificate from the Call Bridge if doing direct federation.</td>
</tr>
<tr>
<td>TURN server</td>
<td>If you configure TLS on your TURN server, then the TURN server will require a certificate/key pair similar to that created for the Web Bridge, so that the WebRTC client trusts the connection. The certificate should be signed by the same Certificate Authority as used for the Web Bridge certificate.</td>
</tr>
</tbody>
</table>

Table 3: Internal CA signed certificates (single split server model)

<table>
<thead>
<tr>
<th>Applications that can use internal CA signed certificate</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Admin</td>
<td>The Meeting Server only allows HTTPS connection to the interface of the Web Admin, so a certificate is required for the Web Admin. Note: The Meeting Server API is routed through the interface of the Web Admin, so a certificate is required even if you configure the Call Bridge through the API rather than the Web Admin Interface.</td>
</tr>
<tr>
<td>Call Bridge</td>
<td>The Web Bridge requires and needs to trust a certificate from the Call Bridge. The Active Directory server also needs to trust a certificate from the Call Bridge. In addition, if your deployment has SIP trunks using TLS, then the Call Bridge requires a certificate for mutual authentication with the SIP call control devices.</td>
</tr>
<tr>
<td>Applications that can use internal CA signed certificate</td>
<td>Reason</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Load Balancer (only for split server deployments and if native Apps are used)</td>
<td>If native Cisco Meeting Apps are deployed then the trunk between the Core and Edge servers needs to authenticate (trust the certificate) presented by the Load Balancer. Note: the external native Cisco Meeting Apps do not check the Load Balancer certificate.</td>
</tr>
<tr>
<td>Trunk (only for split server deployments and if native Apps are used)</td>
<td>If native Cisco Meeting Apps are deployed then the Load Balancer in the Edge server needs to authenticate (trust the certificate) presented by the trunk.</td>
</tr>
<tr>
<td>Recorder</td>
<td>If you enable a Recorder on the Meeting Server, the Call Bridge requires a signed certificate from the Recorder, and the Recorder requires and needs to trust a certificate from the Call Bridge.</td>
</tr>
<tr>
<td>Streamer</td>
<td>If you enable a Streamer on the Meeting Server, the Call Bridge requires a signed certificate from the Streamer, and the Streamer requires and needs to trust a certificate from the Call Bridge.</td>
</tr>
</tbody>
</table>
3 Obtaining certificates

Chapter 1 explains where certificates are required to establish secure connections in the deployment, and the type of certificates required (signed by a public CA or an internal CA). This chapter focuses on how to obtain the different types of certificate, Chapter 4 covers where to install them.

**Note:** If you are connecting a Lync deployment to the Meeting Server, you are advised to use the same Certificate Authority (CA) that is trusted by Lync Front End servers. Contact your Lync adviser for details of the CA and for support on the Meeting Server–Lync integration.

All certificates require you following a 3 step process:

1. Generate a private key and the Certificate Signing Request (.csr) file for the specific Meeting Server component.

   **Note:** The public key is created and held within the .csr file.

2. Submit the .csr file to the CA (public CA or internal CA) for signing.

3. Install the signed certificate and intermediate CA bundle (if any) on the Meeting Server using SFTP.

The remainder of this chapter provides examples for steps 1 and 2. Chapter 4 covers step 3.

**Note:** Instructions for generating self-signed certificates using the Meeting Server’s MMP commands are provided in Chapter 6. These are useful for testing your configuration in the lab. However, in a production environment you are advised to use certificates signed by a Certificate Authority (CA).

**Note:** The Meeting Server supports certificates signed using SHA1 and SHA2 algorithms. When the Meeting Server creates certificate signing requests, they are signed using SHA256 in accordance with rules which CAs now operate under.

### 3.1 Generating a private key and Certificate Signing Request (.csr file)

This section describes using the Meeting Server MMP `pki` command to create a public key and .csr file. If you prefer to use a third party tool to do this, follow the instructions from the third party then resume following this guide from Section 3.2. If you prefer to use OpenSSL to create a private key and .csr file then Appendix A provides an overview of the steps to follow.
You can use the `pki csr <key/cert basename>` command to generate two files: the private key `<basename>.key` and the certificate signing request file `<basename>.csr`. They can be immediately retrieved from the Meeting Server by using SFTP.

**Note:** The basename must NOT include a “.” or “_”, for example `pki csr basename` is valid, but `pki csr base.name` or `pki csr base_name` are not allowed.

To generate the private key and Certificate Signing Request file:

1. Log in to the MMP
2. Type the `pki csr` command using this syntax
   ```
   pki csr <key/cert basename> <CN:value> [OU:<value>] [O:<value>] [ST:<value>] [C:<value>] [subjectAltName:<value>]
   ```

   where

   `<key/cert basename>` is a string identifying the new key and CSR. Can contain alphanumeric, hyphen or underscore characters.

   `CN,OU,O,ST,C,subjectAltName` are described in Table 4. Those marked optional can be omitted if you are creating a certificate request file using the `pki csr` command for signing by a local CA. If you are creating a certificate request file for a public Certificate Authority to sign, you are advised to provide all of the attributes.

**Table 4: Attributes in a .csr file**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Optional/Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>Common Name</td>
<td>Required, see notes below</td>
</tr>
<tr>
<td></td>
<td>This is the fully qualified domain name (FQDN) that specifies the server's exact location in the Domain Name System (DNS). For example, a component with hostname webBridge1 and parent domain example.com has the fully qualified domain name webBridge1.example.com. The FQDN uniquely distinguishes the component from any other components called webBridge1 in other domains.</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>Organization or Business name</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Usually the legal incorporated name of a company. It should include any suffixes such as Ltd., Inc., or Corp. Use “” around the attribute if more than one word, e.g. “Example Inc.”</td>
<td></td>
</tr>
<tr>
<td>OU</td>
<td>Organizational unit or Department name</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>For example, Support, IT, Engineering, Finance. Use “” around the attribute if more than one word, e.g. “Human Resources”</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Location</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>City or town. For example, London, Boston, Milan, Berlin.</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>Province, Region, County or State</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>For example, Buckinghamshire California. Do not abbreviate. Use “” around the attribute if more than one word, e.g. New Jersey</td>
<td></td>
</tr>
</tbody>
</table>
## Attribute Guidelines

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Optional/Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Country</td>
<td>Optional</td>
</tr>
<tr>
<td>An email address</td>
<td>An email address to contact the organization. Usually the email address of the certificate administrator or IT department.</td>
<td>Optional</td>
</tr>
<tr>
<td>SAN</td>
<td>Subject Alternative Name From X509 Version 3 (RFC 2459), SSL certificates are allowed to specify multiple names that the certificate should match. This field enables the generated certificate to cover multiple domains. It can contain IP addresses, domain names, email addresses, regular DNS host names, etc, separated by commas. If you specify this list you must also include the <strong>CN</strong> in this list. Although this is an optional field, the SAN field must be completed in order for XMPP clients to accept a certificate, otherwise the XMPP clients will display a certificate error.</td>
<td>Required if you plan to use a dedicated certificate for the XMPP Server or if a single certificate is to be used across multiple components. See note below</td>
</tr>
</tbody>
</table>

### Points to note:

- If you plan to use a dedicated certificate for the Web Bridge then specify in the CN field, the FQDN that is defined in the DNS A record for the Web Bridge. Failure to specify the FQDN may result in browser certificate errors.

- If you plan to use a dedicated certificate for the XMPP Server then specify in the CN field, the FQDN that is defined in the DNS SRV record for the XMPP Server. In the `subjectAltName` field specify the domain name of the XMPP server and the DNS SRV record for the XMPP Server.

- If you plan to use the same certificate across multiple components, for example the Web Bridge, XMPP Server, Call Bridge, TURN server, Web Admin, Reorder and Streamer then specify your domain name (DN) in the CN field, and in the SAN field specify your domain name (DN) and the FQDN for each of the components that will use the certificate.

- In the SAN field, ensure there are no spaces between the "," delimiter and the items in the list.

For example:

**CN=example.com**

**SAN=callbridge.example.com,xmppserver.example.com,webbridge.example.com,example.com**

If using the `pki csr` command:

`pki csr <key/cert basename> <CN:value> [OU:<value>] [O:<value>] [ST:<value>] [C:<value>] [subjectAltName:value]`

the command is:

`pki csr onecert CN:example.com subjectAltName:callbridge.example.com,xmppserver.example.com,webbridge.example.com`
Note: If you use the `pki` command, the CN is automatically appended to the SAN list, do not list the CN in the SAN list, as shown in the example above.

3.1.1 CSR for the Call Bridge

Depending on how the Call Bridge is used within your deployment, it may need private key/certificate pairs:

- to establish communication with the Web Bridge. It is important for the security of the deployment that configuration is only accepted from a trusted Call Bridge.
- to establish TLS connections with SIP Call Control devices.
- to establish TLS connection with the Lync Front End (FE) server. To ensure that a certificate will be trusted by the Lync FE server:
  
  - the CN in the Certificate must be the same as the FQDN that was added when configuring the Meeting Server as a trusted application and static routes on Lync FE server.
  - if the certificate has a `subjectAltName` list then the FQDN must also be added to the list.
  - sign the certificate using a trusted CA server, such as the CA that has issued the certificates for the Lync FE server.

For example:

```
pki csr callbridge CN:www.example.com O:"Example Inc."
```

or

```
pki csr callbridge CN:www.example.com O:"Example Inc." subjectAltName:callbridge.example.com
```

The example will generate two files: callbridge.key and callbridge.csr. The files can be immediately retrieved from the Meeting Server by using SFTP. Submit the `.csr` file to a public CA for signing, see Section 3.2.

Section 4.7 provides details on uploading certificates for the Call Bridge.

3.1.2 CSR for the Web Bridge

Web browsers look at the `cn` field to determine the Web Bridge’s FQDN. To avoid web browser certificate errors, follow this advice:

- if you are using a dedicated certificate for the Web Bridge: in the `cn` field, specify the FQDN that is defined in the DNS A record for the Web Bridge. Failure to specify the FQDN may result in browser certificate errors. If the `subjectAltName` field is used, then the FQDN that is specified in the `cn` field needs to be included in the `subjectAltName` field if it is not automatically appended. Note: `pki csr` will automatically appended the `cn` to the `san` list if a `san` list exists.
if you plan to use the same certificate across multiple components (Web Bridge, XMPP Server, Call Bridge and TURN server): specify your domain name (DN) in the CN field, and in the SAN field specify your domain name (DN) and the FQDN for each of the components that will use the certificate.

For example:

```
pki csr webbridge CN:www.example.com O:“Example Inc.”
```

or

```
pki csr webbridge CN:www.example.com O:“Example Inc.”
subjectAltName:guest.example.com
```

The example will generate two files: webbridge.key and webbridge.csr. The files can be immediately retrieved from the Meeting Server by using SFTP. Submit the .csr file to a public CA for signing, see Section 3.2.

Section 4.6 provides details on uploading certificates for the Web Bridge.

### 3.1.3 CSR for the XMPP server

Native Cisco Meeting App look at the subjectAltName and CN fields to determine the XMPP server’s domain. To avoid client certificate errors, ensure that the CSR for the XMPP server specifies:

- the DNS record for the XMPP server in the CN field, or in subjectAltName field
- the XMPP server’s domain name in the subjectAltName field

For example, if the XMPP domain is configured as example.com and DNS is xmpp.example.com, the CN should be xmpp.example.com and within the SAN list you must add xmpp.example.com and example.com. Note: \texttt{pki csr} will automatically append the CN to the SAN list if a SAN list exists.

```
pki csr xmppserver CN:xmpp.example.com O:“Example Inc.”
subjectAltName:example.com
```

or using the wildcard:

```
pki csr xmppserver CN:*.@example.com O:“Example Inc.”
subjectAltName:example.com
```

generates two files: xmppserver.key and xmppserver.csr, for the XMPP server in domain example.com. Submit the .csr file to a public CA for signing, see Section 3.2. The files can be immediately retrieved from the Meeting Server by using SFTP.

Section 4.4 provides details on uploading certificates for the XMPP server.

\textbf{Note:} When applying certificates for XMPP multi-domains the same requirements apply; the CSR must include both the DNS record for the XMPP server and the domain names for the XMPP multi-domain.
3.1.4 CSR for trunk/Load Balancer pair

In a single split deployment, the Core server hosting the XMPP service requires a trunk to the Load Balancer on the Edge server. You need to create a private key/certificate pair for the trunk and for the Load Balancer. The trunk and the Load Balancer cannot use the same private key/certificate pair, but their certificates can be signed by an internal CA.

1. Create a private key/certificate pair for the Load Balancer enabled on the Edge server.

   For example:
   ```
pki csr edge1 CN:www.example.com
```
   will create edge1.key and edge1.csr.

2. Create a private key/certificate pair for the Core server with the XMPP service enabled.

   For example:
   ```
pki csr core1 CN:www.example.com
```
   will create core1.key and core1.csr.

3. Use an Internal CA to sign edge1.csr and core1.csr certificate signing request files and obtain the corresponding edge1.crt and core1.crt certificates, as well as the Internal CA certificate (bundle). See Section 3.3.

   Section 4.5 provides details on uploading certificates for the trunk and Load Balancer.

3.1.5 CSR for the TURN server

If you plan to use TLS on a TURN server, then the TURN server will require a certificate/key pair similar to that created for the Web Bridge, so that WebRTC clients trusts the connection. The certificate should be signed by the same Certificate Authority as used for the Web Bridge certificate.

For example:
```
pki csr turnserver CN:www.example.com O:“Example Inc.”
```

The example will generate two files: turn.key and turn.csr. The files can be immediately retrieved from the Meeting Server by using SFTP. Submit the .csr file to a public CA for signing, see Section 3.2.

Section 4.8 provides details on uploading certificates for the TURN server.

3.2 Signing the CSR using a public Certificate Authority

Refer to Section 1.2 for a list of public CA signed certificates required for the Meeting Server.

To obtain a public CA signed certificate, send the generated .csr file to your preferred Certificate Authority, for example Verisign. The CA will verify your identity and issue a signed certificate. The certificate file will have a .pem, .cer or .crt extension. Appendix B provides a brief overview of file extensions used for certificate files.
Before transferring the signed certificate and the private key to the Meeting Server, check the certificate file. If the CA has issued you a chain of certificates, you will need to extract the certificate from the chain. Open the certificate file and copy the specific certificate text including the BEGIN CERTIFICATE and END CERTIFICATE lines and paste into a text file. Save the file as your certificate with a .crt, .cer or .pem extension. Copy and paste the remaining certificate chain into a separate file, naming it clearly so you recognize it as an intermediate certificate chain and using the same extension (.crt, .cer or .pem). The intermediate certificate chain needs to be in sequence, with the certificate of the CA that issued the chain first, and the certificate of the root CA as the last in the chain.

Go to Chapter 4 for information on installing signed certificates and private keys on the Meeting Server.

**Note:** In deployments where the Meeting Server is trunked to Cisco Unified Communications Manager, Cisco Unified Communications Manager requires the Call Bridge certificate to be signed using a template that allows for an Extended Key Usage containing both TLS Web Client Authentication and TLS Web Server Authentication.

### 3.3 Signing the CSR using an internal Certificate Authority

Refer to section Section 1.2 for a list of internal CA signed certificates required for the Meeting Server.

**Note:** In deployments where the Meeting Server is trunked to Cisco Unified Communications Manager, Cisco Unified Communications Manager requires the Call Bridge certificate to be signed using a template that allows for an Extended Key Usage containing both TLS Web Client Authentication and TLS Web Server Authentication. Microsoft Active Directory Certificate Services can issue this type of certificate.

This section applies if you are using Microsoft Active Directory as an internal CA. If you are using a different internal CA, please follow the corresponding instructions, and then resume following this guide from Chapter 4.

To obtain an internal CA signed certificate, follow these steps:

1. Transfer the generated .csr file to the CA, for example an Active Directory server with the Active Directory Certificate Services Role installed.

2. Issue the following command in the command line management shell on the CA server replacing the path and CSR filename with your information:
   ```
   certreq -Submit -attrib "CertificateTemplate:WebServer" <path\csrfilename>
   ```
   For example:
   ```
   certreq -Submit -attrib "CertificateTemplate:WebServer"
   C:\Users\Administrator\Desktop\example.csr
   ```
3. After entering the command, a CA selection list is displayed similar to that below. Select the correct CA and click OK.

![Certification Authority List]

If your Windows account has permission to issue certificates, you will be prompted to save the resulting certificate. Save the file with a .crt, .cer or .pem extension, for example example.crt. Go to step 4. See Appendix B for a brief overview of certificate file extensions.

If you do not see a prompt to issue the resulting certificate, but instead see a message in the command prompt window that the 'Certificate request is pending: taken under submission', and listing the Request ID, then make a note of the RequestID.

```
C:\Users\Administrator>certreq -submit -attrib "CertificateTemplate:WebServer" C:\\Users\\Administrator\\Desktop\\demokitcer.pem Active Directory Enrollment Policy  
{6BD5DDB7-571F-4C77-AFEG-5C6E47BF77D5} 
RequestId: 8  
RequestId: 8' 
Certificate request is pending: Taken Under Submission (0)
```

Follow these steps to obtain the issued certificate.

a. Using the Server Manager page on the CA, locate the Pending Requests folder under the CA Role.

b. Right-click on the pending request that matches the Request ID given in the cmd window and select All Tasks > Issue.

c. The resulting signed certificate will be in the Issued Certificates folder. Double-click on the certificate to open it and open the Details tab.
d. Click **Copy to File** which will start the Certificate Export Wizard.

e. Select **Base-64 encoded X.509 (.CER)** and click **Next**.

f. Browse to the location in which to save the certificate, enter a name for example **xmpp** and click **Next**.

g. Save the resulting certificate with a .crt, .cer or .pem extension, for example **xmpp.crt**

4. Go to Chapter 4 for information on installing signed certificates and private keys on the Meeting Server.
4 Installing signed certificates and private keys on the Meeting Server

To summarize Chapter 1, the single split Meeting Server deployment requires public CA signed certificates for:

- **the Web bridge**, if Web RTC clients are to be enabled for use by end users. The Web RTC client requires a public CA signed certificate from the Web Bridge in order to trust the connection.
- **the TURN server**, if you plan to use TLS connections for secure communication.
- **the XMPP server**, if native Cisco Meeting Apps (PC, Mac, iOS) are to be used by end users. The Native Cisco Meeting Apps require a public CA signed certificate from the XMPP server in order to trust the connection.
- **the Call Bridge**, if direct Lync federation over a public network is required. The Lync Edge server requires a public CA signed certificate from the Call Bridge in order to trust the connection.

And internal CA signed certificates for:

- **the Web Admin**. The Meeting Server API is routed through the Web Admin Interface, so a certificate is required even if you configure the Call Bridge through the API rather than the Web Admin Interface.

  **Note**: this guide assumes that you have already installed the private key/certificate pair for the Web Admin Interface as described in the Meeting Server Installation Guide. If you have not, do so now.

- **the Call Bridge**. The Web Bridge requires a certificate from the Call Bridge. The Active Directory Server also requires a certificate from the Call Bridge. In addition, if your deployment has SIP trunks, then the Call Bridge requires a certificate for mutual authentication with the SIP call control devices.

- **the Load Balancer**, in split server deployments and if native Cisco Meeting Apps (PC, Mac, iOS) are to be used by end users. The trunk between the Core and Edge server needs to authenticate (trust) the certificate presented by the Load Balancer. This is part of the mutual authentication that needs to be set up between the Load Balancer and the trunk so that the XMPP server trusts the connection. The Load Balancer and the trunk cannot use the same certificate. Note: the external native Cisco Meeting Apps do not check the Load Balancer certificate.
the Trunk, in split server deployments and if native Cisco Meeting Apps (PC, Mac, iOS) are to be used by end users. The Load Balancer in the Edge server needs to authenticate (trust) the certificate presented by the trunk. This is the other part of the mutual authentication that needs to be set up between the Load Balancer and the trunk. The trunk and the Load Balancer cannot use the same certificate.

- If you enable a Recorder on the Meeting Server, the Call Bridge requires a certificate from the Recorder, and the Recorder requires and needs to trust a certificate from the Call Bridge. Refer to the Recorder section in the Cisco Meeting Server Single Split deployment guide for details on uploading the certificates and configuring the Recorder.

- If you enable a Streamer on the Meeting Server, the Call Bridge requires a certificate from the Streamer, and the Streamer requires and needs to trust a certificate from the CallBridge. Refer to the Streamer section in the Cisco Meeting Server Single Split deployment guide for details on uploading the certificates and configuring the Streamer.

4.1 Reusing a private key and certificate

You do not need to have a different private key/certificate pair for each certificate install. In some circumstances you can copy and reuse the private key and certificate for multiple services. Here is some advice if you reuse a private key/certificate pair:

- if you are connecting a Lync deployment to your Meeting Server, you are advised to use the Certificate Authority (CA) trusted by the Lync deployment.
- use filenames for the certificate and private key that reflect where they are used, for example: wewebadmin.crt and wewebadmin.key.
- do not use the same private key/certificate pair between a trunk and Load Balancer, they require different private key/certificate pair, see Section 3.1.4.

4.2 Uploading the Private Keys and Certificates to the MMP

1. SSH into the MMP, and login
2. Use SFTP to upload each private key/certificate pair and certificate bundle
3. Use the MMP PKI command: pki list to check which files have been uploaded. pki list will also list any SSH keys and CSR files uploaded to the MMP.

Note: Private keys and certificates must NOT include a “.” within the filenames except immediately before the file extension. For example callbridge.key is valid, but call.bridge.key is not allowed.
4.3 Inspecting a file type and checking that a certificate and private key match

Before installing a private key/certificate pair on the Meeting Server, make sure that you have the correct files to install. This section provides a brief overview of using the MMP commands: pki inspect, pki match, and pki verify, to check the identity of the files you plan to install.

To inspect a file to determine whether it is still valid (expiry date):

```
pki inspect <filename>
```

To check that a certificate matches a private key:

```
pki match <keyfile> <certificatefile>
```

To check that a certificate is signed by the CA and that the certificate bundle can be used to assert this:

```
pki verify <cert> <certbundle/CAcert>
```

For example:

1. SSH into the MMP, and login
2. Enter the command:
   
   ```
pki inspect xmppserver.crt
   ```
   
   to inspect the contents of the file, for instance to see whether a certificate is still valid.

3. Enter the command:

   ```
pki match xmppserver.key xmppserver.crt
   ```
   
   to check that the file `xmppserver.key` matches file `xmppserver.crt` and together they form one usable identity.

4. Enter the command:

   ```
pki verify xmppserver.crt xmppbundle.crt
   ```
   
   to check that xmppserver.crt is signed by a trusted CA, with the chain of trust established through the chain of intermediate certificates in xmppbundle.crt.

4.4 Installing the Certificate and Private Key for the XMPP Server

If your Meeting Server supports end users using the native Cisco Meeting Apps for PC, Mac, and iOS devices, then you need to install a public CA signed certificate for the XMPP server. The certificate will be used by the Meeting Apps in the initial setting up of the connection to determine whether they can trust the connection with the XMPP server.

The steps below assume that you have already configured the network interface that the XMPP server will use to listen. Refer to the Single Split Server Deployment Guide for information on setting the interface using the `listen` MMP command, before assigning the certificate.
1. SSH into the MMP
2. Disable the XMPP server interface before assigning the certificate
   ```
   xmpp disable
   ```
3. Assign the private key/certificate pair using the command:
   ```
   xmpp certs <keyfile> <certificatefile> [cert-bundle]
   ```
   where `keyfile` and `certificatefile` are the filenames of the matching private key and certificate. If your CA provides a certificate bundle then also include the bundle as a separate file to the certificate.

   For example:
   ```
   xmpp certs xmppserver.key xmppserver.crt xmppserverbundle.crt
   ```
4. Re-enable the XMPP server interface
   ```
   xmpp enable
   ```

If the certificate installs successfully on the XMPP server, the following is displayed:

SUCCESS: Domain configured
SUCCESS: Key and certificate pair match
SUCCESS: license file present
SUCCESS: XMPP server enabled

If the certificate fails to install, the following error message is displayed:

FAILURE: Key and certificate problem: certificate and key do not match
FAILURE: XMPP server configuration not complete

If there is a problem with the certificate bundle, an error message similar to the following is displayed:

SUCCESS: Domain configured
SUCCESS: Key and certificate pair match
FAILURE: certificate verification error: depth=x issuer= x = xx, ST = xxxxxx, L = xxxxxx, O = xxxxx, OU = xxxxxx, CN = xxxxxxxxxx, emailAddress = xxxxxxxxxx@xxxxx.xxx Verification error: unable to get issuer certificate Failed cert:
Certificate:
   Data:
   Version: xx
   Signature Algorithm: sha1WithRSAEncryption
   Issuer: C=xx, ST= xxxxxx, L= xxxxxx, O= xxxxx, OU= xxxxxx, CN= xxxxxxxxxx, emailAddress = xxxxxxxxxx@xxxxx.xxx
   Validity
   Not Before: <month><time><year>
   Not After: <month><time><year>
   Subject: C=xx, O=xxx, OU=xxxxxxxxxxx, CN=xxxxxxxxxxxxxxxxx
   Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
SUCCESS: license file present
FAILURE: XMPP server configuration not complete

Check the certificate bundle to ensure that there is no break in the certificate chain.

**Note:** The XMPP license is included in the Cisco Meeting Server software. See the Single Split Server Deployment Guide for more information on configuring the XMPP server.

### 4.5 Installing the Certificate and Private Keys for the Core to Edge Trunks

This section only needs to be followed if your Meeting Server connects to native Cisco Meeting Apps (PC Client, Mac Client or iOS Client) and you have deployed split Core and Edge servers.

To establish mutual authentication between the Load Balancer in the Edge server and the XMPP service in the Core server you need to:

- on the Load Balancer, install the private key/certificate pair for the Load Balancer, and the certificate for the trunk
- on the trunk, install the private key/certificate pair for the trunk, and the certificate for the Load Balancer.

**Note:** Do not reuse the Load Balancer certificate for the trunk, or vice versa. You must create different certificates for the Load Balancer and for the trunk.

This will ensure that the trunk and the Load Balancer authenticate each other’s certificate, creating a secure TLS trunk between the Load Balancer and the XMPP server.

**Note:** This guide assumes that you have already set up the network interface for the Load Balancer and the trunk interface on the Meeting Server, if not follow the steps in the Single Split Server Deployment Guide, before assigning the certificate.

1. SSH into the MMP of the Edge server
2. Create an Edge instance, using the command:

   ```
   loadbalancer create <tag>
   ```

   If the tag for the Edge server is "Edge1toLB", type:

   ```
   loadbalancer create EdgeltoLB
   ```

3. Assign the private key/certificate pair to the Load Balancer and the trunk’s certificate using the command:

   ```
   loadbalancer auth <tag> <keyfile> <certificatefile> <trust-bundle>
   ```
where `keyfile` and `certificatefile` are the filenames of the matching private key/certificate pair for the Load Balancer, and `<trust-bundle>` is the certificate for the trunk.
For example:

```
loadbalancer auth EdgeltoLB edgel.key edgel.crt core1.crt
```

**Note:** The trunk certificate `core1.crt` is added as a ‘trust bundle’ to the Edge server

4. Configure the trunk interface and port, using:

```
loadbalancer trunk <tag> <iface>:<port>
```
for example, if the trunk connection will be allowed on interface A, port 4999, then type:

```
loadbalancer trunk EdgeltoLB a:4999
```

5. Configure the public interface and port (for accepting client connections), using:

```
loadbalancer public <tag> <iface:port whitelist>
```
for example, if client connections are to be allowed on B, port 5222, then type:

```
loadbalancer public EdgeltoLB b:5222
```

6. In a common Edge server deployment, the Web Bridge is also enabled and needs to make use of the trunk. To allow this, configure the loopback as a public interface, e.g.

```
loadbalancer public EdgeltoLB b:5222 lo:5222
```

7. Enable the trunk, using:

```
loadbalancer enable <tag>
```
for example

```
loadbalancer enable EdgeltoLB
```

**Note:** The public port is not opened until there is a trunk to service the connection.

8. SSH into the MMP of the Core server

9. Create a trunk between the Core and Edge server for xmpp traffic

```
trunk create <tag> <port/service name>
```
For example:

```
trunk create trunktoEdgel xmpp
```

10. Assign the private key/certificate pair to the trunk and the Load Balancer’s certificate using the command:

```
trunk auth <tag> <key-file> <cert-file> <trust-bundle>
```
where `keyfile` and `certificatefile` are the filenames of the matching private key/certificate pair for the trunk, `<trust-bundle>` is the certificate for the Load Balancer.
For example:

```
trunk auth trunktoEdgel core1.key core1.crt edgel.crt
```
11. Configure the Edge server that this trunk will connect to, using:

```bash
trunk edge <tag> <edge name/ip address> [default port]
```

For example, if the Edge server name is edge1.example.com using port 4999, then type:

```bash
trunk edge trunktoEdge1 edge1.example.com 4999
```

**Note:** If the domain name resolves to multiple IP addresses, a connection will be attempted to all.

12. Enable the trunk interface

```bash
trunk enable <tag>
```

For example:

```bash
trunk enable trunktoEdge1
```

**Note:** To see the full list of Load Balancer and trunk commands, see the Cisco Meeting Server MMP Command Reference.

### 4.6 Installing the Certificate and Private Key for the Web Bridge

If your Meeting Server supports end users using the WebRTC App then you need to install a public CA signed certificate for the Web Bridge. The certificate will be used by browsers when determining whether to trust the connection with the Web Bridge.

The steps below assume that you have already configured the DNS record and the network interface that the Web Bridge will use to listen. Refer to the Single Split Server Deployment Guide for information on setting the interface using the `listen` MMP command before assigning the certificates.

For each Web Bridge:

1. SSH into the MMP
2. Disable the Web Bridge interface before assigning the certificate
   ```bash
   webbridge disable
   ```
3. Assign the private key/certificate pairs using the command:
   ```bash
   webbridge certs <keyfile> <certificatefile> [<cert-bundle>]
   ```
   where `keyfile` and `certificatefile` are the filenames of the matching private key and certificate. If your CA provides a certificate bundle then also include the bundle as a separate file to the certificate.
For example:

```
callbridge certs callbridge.key callbridge.crt callbridgebundle.crt
```

4. Re-enable the Web Bridge interface

```
webbridge enable
```

If the certificate installs successfully on the Web Bridge, then the following is displayed:

SUCCESS: Key and certificate pair match
SUCCESS: Webbridge enabled

If the certificate fails to install, the following error message is displayed:

FAILURE: Key and certificate problem: certificate and key do not match
FAILURE: Webbridge configuration not complete

**Note:** Use the MMP command `webbridge certs none` to remove the certificate configuration from the Web Bridge.

### 4.7 Installing the Certificate and Private Key for the Call Bridge

As explained in **Section 3.1.1**, depending on how the Call Bridge is used within your deployment, it may need private key/certificate pairs.

The steps below assume that you have already configured the network interface that the Call Bridge will use to listen. Refer to the Single Split Server Deployment Guide for information on setting the interface using the MMP command `listen` before assigning the certificates.

For each Call Bridge:

1. SSH into the MMP of the Meeting Server.
2. Assign the private key/certificate pairs using the command:

```
callbridge certs <keyfile> <certificatefile>[<cert-bundle>]
```

where **keyfile** and **certificatefile** are the filenames of the matching private key and certificate. If your CA provides a certificate bundle then also include the bundle as a separate file to the certificate.

For example:

```
callbridge certs callbridge.key callbridge.crt callbridgebundle.crt
```

3. Restart the Call Bridge interface to apply the changes.

```
callbridge restart
```

If the certificate installs successfully on the Call Bridge, then the following is displayed:

SUCCESS: listen interface configured
SUCCESS: Key and certificate pair match

If the certificate fails to install, the following error message is displayed:

FAILURE: Key and certificate problem: certificate and key do not match
4.7.1 Establishing Trust between the Call Bridge and the Web Bridge

The Web Bridge allows configuration of guest logins and image customizations to be pushed from a Call Bridge (see the Customization Guidelines document). It is important for the security of the deployment that configuration is only accepted from the trusted Call Bridge.

Trust between the Call Bridge and the Web Bridge is established by providing the Web Bridge with the public certificate of the Call Bridge. The Web Bridge can use this to challenge the Call Bridge to prove by cryptographic means that it is the owner of the certificate. If the Call Bridge cannot prove that it is the owner of one of the trusted certificates, the Web Bridge will not accept configuration.

**Note:** In split deployments the Call Bridge certificate needs to be copied from the Core server to the Edge server before you use the `webbridge trust <callbridgecert>` command on the Edge server. On the Core server running Call Bridge:

```
cms> callbridge
Listening interfaces : a
Key file : callbridge.key
Certificate file : callbridge.cer
```

Use SFTP to copy the certificate and key file from the Core server to the Edge server running Web Bridge. Then on the Edge server add the certificate to the Web Bridge trust store.

To add the Call Bridge certificate to the Web Bridge trust store:

1. Check which certificate the Call Bridge is using by issuing the `callbridge` command
2. Disable the Web Bridge
3. Add the Call Bridge certificate to the trust store using the command:

   ```
   webbridge trust <callbridgecert|cert-bundle>
   ```

   ```
   cms> webbridge disable
   cms> webbridge trust callbridge.crt
   cms> webbridge enable
   SUCCESS: Key and certificate pair match
   SUCCESS: webbridge enabled
   ```

4. Re-enable the Web Bridge
5. To verify that the Web Bridge has the Call Bridge certificate in its trust store:
4.8 Installing the Certificates and Private Keys for TURN Server

If you plan to use TLS for secure communication, then you need to install a signed certificate for the TURN server, using the same CA to sign the certificate as that used for the Web Bridge. The certificate will be used by browsers when determining whether to trust the connection with the Meeting Server.

The steps below assume that you have already configured the network interface that the TURN server will use to listen. Refer to the Single Split Server Deployment Guide for information on setting the interface using the `listen` MMP command before assigning the certificates.

For each TURN server:

1. SSH into the MMP of the host server
2. Disable the TURN server interface before assigning the certificate
   
   ```
   turn disable
   ```
3. Upload the signed certificate and intermediate CA bundle (if any) to the Meeting Server using SFTP.
4. Check that the certificate (and certificate bundle) and the private key match
   
   ```
   pki verify <certificate> <cert bundle/CA cert> [<CA cert>]
   ```
5. Assign the certificate (and certificate bundle) and private key pair to the Turn server
   
   ```
   turn certs <keyfile> <certificatefile> [cert-bundle]
   ```
   where keyfile and certificatefile are the filenames of the matching private key and certificate. If your CA provides a certificate bundle then also include the bundle as a separate file to the certificate.
   For example
   
   ```
   turn certs turn.key turn.crt turnbundle.crt
   ```
6. Re-enable the TURN server
   
   ```
   turn enable
   ```

4.9 TLS Certificate Verification

You can enable Mutual Authentication for SIP and LDAP in order to validate that the remote certificate is trusted. When enabled, the Call Bridge will always ask for the remote certificate.
(irrespective of which side initiated the connection) and compare the presented certificate to a trust store that has been uploaded and defined on the server.

The MMP commands available are:

- to display the current configuration
  
  `tls <sip|ldap>`

- to define the Certificate Authorities to be trusted
  
  `tls <sip|ldap> trust <crt bundle>`

- to enable/disable certificate verification or whether OCSP is to be used for verification
  
  `tls <sip|ldap> verify enable|disable|ocsp`

See the [MMP Command Reference](#) guide for further information.
5 Troubleshooting problems with certificates

This section covers a few common troubleshooting problems. Refer to the Meeting Server Knowledgebase for further Frequently Asked Questions relating to certificates.

5.1 Warning message that service is untrusted

The message is displayed if:

- you have used an internal CA which is not in your trust store,
- you have used a self-signed certificate where a public or internal CA signed certificate is required. Re-issue the certificate and have it signed by a trusted CA: this can be an internal CA unless you want public access to this component.

5.2 Client certificate error

In order for the clients to communicate with the XMPP server they must trust the connection. For the client to trust the connection, and avoid client certificate errors, the certificate for the XMPP server should include:

- the DNS record for the XMPP server in the CN field,
- the XMPP domain name in the subjectAltName field

For example, if clients are using 'firstname.surname@example.com', then the XMPP domain is example.com and DNS is xmpp.example.com. The CN should specify xmpp.example.com and within the SAN list you must add example.com. The CN is automatically appended to the SAN list.

```
pki csr xmppserver CN:xmpp.example.com O:“Example Inc.”
sanAltName:example.com
```

or using the wildcard:

```
pki csr xmppserver CN:*example.com O:“Example Inc.”
sanAltName:example.com
```

5.3 Browser certificate error

In order for the WebRTC clients to communicate with the Web Bridge they must trust the connection. For the client to trust the connection, the certificate for the Web Bridge must include the Fully Qualified Domain Name (FQDN) that specifies the server’s exact location in the Domain Name system (DNS). It will have been defined in the DNS A record for the Web Bridge
service. Failure to specify the FQDN in the Web Bridge certificate will result in browser certificate errors.

In addition, if TLS is configured on the Meeting Server, then the TURN server requires a similar certificate to the certificate/key pair assigned to the Web Bridge. Failure to assign a suitable certificate/key pair to the TURN server may result in the client not trusting the connection and the browser displaying a certificate error.

5.4 Call Bridge cannot connect to Web Bridge

Either the Web Bridge does not have the certificate of the Call Bridge in its trust store or the certificate has expired (Authentication error).

Type `webbridge` to show the certificate that is in the Web Bridge’s trust bundle.

Type `pki inspect <certificate name>` to show you the validity of the certificate.

5.5 Using the same private key/certificate pair between trunk and Load Balancer.

The trunk and Load Balancer will be unable to authenticate each other as the private key/certificate pair is the same on both sides (even using trust bundles). Create a different private key/certificate pair for the trunk and Load Balancer.

5.6 Problem connecting to Lync Front End server

Check that the CA that signed the Call Bridge certificate is the same CA that was used to sign the certificate for the Lync Front End server. If the Call Bridge certificate is not signed by the same CA that was used to sign the certificate for the Lync Front End server, then make sure the Lync server can trust the Call Bridge certificate by uploading the Call Bridge Trusted CA certificate to the Root Trust Store of the Lync Servers.

Ensure that the FQDN that was added on Lync, is also present as the CN in the Call Bridge’s certificate.
6  Creating and using certificates in a test environment

You can create a private key and self-signed certificate on the Meeting Server using the `pki selfsigned` command.

Self-signed certificates cannot be used for Lync authentication (as CA is not a trusted authority). But self-signed certificates can be used for Web Admin, for trunk/Load Balancer, and mutual authentication between the Call Bridge and Web Bridge, although the browser will display a certificate error. It is strongly recommended that you use self-signed certificates in a test environment, rather than in a production environment.

To generate a local private key and a self-signed certificate on the Meeting Server:

1. Log in to the MMP and type the command:

   `pki selfsigned <key/cert basename>`

   where `<key/cert basename>` identifies the key and certificate which will be generated.

   For example:

   `pki selfsigned callbridge`

   creates a local private key named `callbridge.key` and a self-signed certificate named `callbridge.crt`
Appendix A  OpenSSL Commands for Generating Certificates

Instead of the MMP pki command described in Chapter 3, OpenSSL can be used to generate private keys, certificate signing requests and certificates. This appendix details the OpenSSL commands to use. The examples assume OpenSSL is running on Windows, although OpenSSL can be used on other platforms.

**Note:** Run OpenSSL in Administrator mode.

### A.1 Generating RSA private keys and CSR files

Use the OpenSSL toolkit on your computer.

To generate a new RSA private key and CSR file, use the command:

```
openssl req -out <certname>.csr -new -newkey rsa:2048 -nodes -keyout <keyname>.key
```

For example:

```
openssl req -out webbridge.csr -new -newkey rsa:2048 -nodes -keyout webbridge.key
```

generates a CSR file named webbridge.csr and an RSA 2048 bit private key named webbridge.key,

**Note:** The keyname and certname must NOT include a “.” or “_”, for example `webbridge` is valid, but `web.bridge` or `web_bridge` are not allowed.

To generate a certificate signing request (CSR) for an existing private key, use the command:

```
openssl req -out <certname>.csr -key <keyname>.key -new
```

For example:

```
openssl req -out xmppserver.csr -key xmppserver.key -new
```

generates a CSR file named xmppserver.csr based on an existing private key named xmppserver.key,

If you intend to self-sign the certificate using OpenSSL, then no intermediate CSR file is required, go to the next section.

### A.2 Signing CSR files

To sign the CSR using a public CA follow the instructions in Section 3.2.
To sign the CSR using an internal CA follow the instructions in Section 3.3.

To self-sign the certificate, use the OpenSSL command:

```
openssl req -x509 -nodes -days 100 -newkey rsa:2048 -keyout <keyname>.key -out <certname>.crt
```

For example:

```
openssl req -x509 -nodes -days 100 -newkey rsa:2048 -keyout callbridge.key -out callbridge.crt
```

generates a new private key named callbridge.key and a (final) certificate named callbridge.crt.

### A.3 Creating Certificates for Database Clustering

This section details how to create certificates for database clustering using OpenSSL commands.

The certificates created for database clustering must be signed by the same Certificate Authority (CA). Because the database clustering is not user-accessible, the CA, keys and certificates can be generated internally using OpenSSL.

**Note:** Ensure that you run OpenSSL with administrator privileges.

**Note:** The keyname and certname must NOT include a “.” or “_”, for example `db01_ca` or `db01.ca` are not allowed.

Follow these steps:

1. Define a CA, and create the private/public key pair and certificate for the CA.
   
   a. Generate a private key and certificate request (.csr) pair for a CA defined by you. Use this OpenSSL syntax

   ```
   openssl req -new -text -nodes -keyout <keyname>.key -out <certname>.csr -subj /C=<country>/ST=<state>/L=<location> /O=<organization>/OU=<organizational unit>/CN=<authorityname>
   ```

   For example:

   ```
   openssl req -new -text -nodes -keyout db01ca.key -out db01ca.csr -subj /C=UK/ST=London/L=London/O=Example
   ```

   creates the private key `db01ca.key` and the certificate signing request file `db01ca.csr` for the CA defined in the attributes following `-subj`

   b. Create a certificate for the CA, using the private key and certificate request (.csr) generated in step 1a.
openssl req -x509 -text -in db01ca.csr -key db01rca.key -out db01ca.crt -days 3650
creates the certificate db01ca.crt

2. Use the CA credentials generated in step 1 to output a private key and signed certificate for the database server and database client.

   a. Generate a private key and certificate request (.csr) for the database server:

      openssl req -new -nodes -keyout <keyname>.key -out <certname>.csr -subj /C=<country>/ST=<state>/L=<locality>
      /O=<organization> /OU=<organizational unit>/CN=<nodename>

      where nodename is the actual name of the server hosting the database. For example:

      openssl req -new -nodes -keyout db01server.key -out db01server.csr -subj /C=UK/ST=London/L=London/O=Example
      creates the key db01server.key and the certificate signing request file db01server.csr

   b. Generate the CA signed certificate for the database. For example:

      openssl x509 -req -CAcreateserial -in db01server.csr -CA db01ca.crt -CAkey db01ca.key -out db01server.crt -days 3650

      creates the certificate db01server.crt

   c. Generate a private key and certificate request (.csr) for the database. The CommonName (CN) for a database client must equal “postgres”.

      openssl req -new -nodes -keyout <keyname>.key -out <certname>.csr
      -subj /C=<country>/ST=<state>/L=<locality>/O=<organization>
      /OU=<organizational unit>/CN=postgres

      For example:

      openssl req -new -nodes -keyout db01client.key -out db01client.csr -subj /C=UK/ST=London/L=London/O=Example
      /OU=/CN=postgres

      creates the key db01client.key and the certificate signing request file db01client.csr.

   d. Generate the CA signed certificate for the database client. For example:

      openssl x509 -req -CAcreateserial -in db01client.csr -CA db01ca.crt -CAkey db01ca.key -out db01client.crt -days 3650

      creates the certificate db01client.crt

3. Follow the steps in Section 4 to upload and assign the database certificates and private keys.
a. Each server hosting a database the following keys and certificates to be uploaded:
   ○ database cluster server certificate (generated in step 2)
   ○ database cluster server key (generated in step 2)
   ○ database cluster client certificate (generated in step 2)
   ○ database cluster client key (generated in step 2)
   ○ database cluster CA certificate bundle (generated in step 1)

b. Each Call Bridge NOT co-located with a database, requires the following keys and certificates to be uploaded:
   ○ database cluster client certificate (generated in step 2)
   ○ database cluster client key (generated in step 2)
   ○ database cluster CA certificate bundle (generated in step 1)

A.4 Installing Certificate and Private Key Pairs

For details on installing the certificate and private key pairs on the Meeting Server, follow the instructions in Chapter 4.
Appendix B  Permitted extensions for certificate files and private keys

The following tables list the permitted file extensions for certificate files and private keys.

### Table 5: Permitted extensions for certificate files

<table>
<thead>
<tr>
<th>Extension</th>
<th>Information on file type</th>
</tr>
</thead>
<tbody>
<tr>
<td>.pem</td>
<td>PEM is both an encoding (ASCII base64) and used as a file extension. Typically imported from a Unix-based Apache Web server and compatible with OpenSSL applications. PEM certificate files are generated automatically. Some secure websites may ask users to upload a PEM file (possibly sent in an e-mail) in order to authenticate their identity.</td>
</tr>
<tr>
<td>.der</td>
<td>Distinguished Encoding Rules (DER) is both an encoding and used as a file extension. Contains a binary representation of the certificate created in the DER format. Commonly used for storing X.509 certificates in public cryptography.</td>
</tr>
<tr>
<td>.cer</td>
<td>Security file provided by a third party Certificate Authority, such as VeriSign or Thwate, that verifies the authenticity of a website. Installed on a Web server to establish the validity of a specific website hosted on the server. The certificates may be encoded as binary DER or as ASCII (Base64) PEM.</td>
</tr>
<tr>
<td>.crt</td>
<td>Certificate used by secure websites (beginning with &quot;https://&quot;) to verify their authenticity. Distributed by companies such as Verisign and Thawte. The certificates may be encoded as binary DER or as ASCII (Base64) PEM. Certificate files are automatically recognized by Web browsers when a user visits a secure site. The information stored in the certificate can be viewed by clicking the lock icon within the browser window.</td>
</tr>
</tbody>
</table>

### Table 6: Permitted extensions for private key files

<table>
<thead>
<tr>
<th>Extension</th>
<th>Information on file type</th>
</tr>
</thead>
<tbody>
<tr>
<td>.key</td>
<td>Used both for public and private PKCS#8 keys. The keys may be encoded as binary DER or as ASCII PEM.</td>
</tr>
<tr>
<td>.pem</td>
<td>Indicates key was encoded using PEM (ASCII base64).</td>
</tr>
<tr>
<td>.der</td>
<td>Indicates key was encoded using binary DER.</td>
</tr>
</tbody>
</table>
## Appendix C  MMP PKI commands

Table 7: List of MMP pki commands

<table>
<thead>
<tr>
<th>Command/ Examples</th>
<th>Description/Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pki</td>
<td>Displays current PKI usage.</td>
</tr>
<tr>
<td>pki list</td>
<td>Lists PKI files i.e. private keys, certificates and certificate signing requests (CSRs).</td>
</tr>
<tr>
<td>pki inspect &lt;filename&gt;</td>
<td>Inspect a file and shows whether the file is a private key, a certificate, a CSR or unknown. In the case of certificates, various details are displayed. If the file contains a bundle of certificates, information about each element of the bundle is displayed. Both PEM and DER format files are handled.</td>
</tr>
<tr>
<td>pki match &lt;key&gt; &lt;certificate&gt;</td>
<td>This command checks whether the specified key and a certificate on the system match. A private key and a certificate are two halves of one usable identity and must match if they are to be used for a service e.g. XMPP.</td>
</tr>
<tr>
<td>pki verify &lt;cert&gt; &lt;cert bundle/CA cert&gt;</td>
<td>This command checks whether the specified certificate is signed by the root CA using the certificate bundle to determine the chain of trust.</td>
</tr>
<tr>
<td>pki verify server.crt bundle.crt</td>
<td></td>
</tr>
<tr>
<td>pki unlock &lt;key&gt;</td>
<td>Private keys are often provided with password-protection. To be used in the Meeting Server, the key must be unlocked. This command prompts for a password to unlock the target file. The locked name will be replaced by an unlocked key with the same name.</td>
</tr>
<tr>
<td>pki csr &lt;key/cert basename&gt; [&lt;attribute&gt;:&lt;value&gt;]</td>
<td>For users happy to trust that Cisco meets the requirements for the generation of private key material, the pki csr command can be used to create private keys and associated Certificate Signing Requests.</td>
</tr>
<tr>
<td>pki csr example CN:www.example.com OU:“My Desk” O:“My Office” L:“Round the corner” ST:California C:US subjectAltName:example.com, <a href="mailto:example@example.com">example@example.com</a></td>
<td>&lt;key/cert basename&gt; is a string identifying the new key and CSR (e.g. “new” results in “new.key” and “new.csr” files) Attributes for the CSR are specified in pairs with the attribute name and value separated by a colon (“:”). Use “ ” around attributes that are more than one word, e.g. O:“Example Inc.”</td>
</tr>
</tbody>
</table>

continued over the page
Attributes are:
CN: commonName which should be on the certificate. The commonName should be the DNS name for the system.
OU: Organizational Unit
O: Organization
L: Locality
ST: State
C: Country
e-mail address
subjectAltName:

Note: Use "" around attributes that are more than one word, e.g. O:"Example Inc."
The CSR file can be downloaded by SFTP and given to a certificate authority (CA) to be signed. On return it must be uploaded via SFTP. It can then be used as a certificate.

The **pki csr <key/cert basename> [attribute]:<value>** accepts subjectAltName as an attribute. IP addresses and domain names are supported for subjectAltName in a comma separated list.

For example:
```
pki csr test1 CN:example.exampledemo.com subjectAltName:exampledemo.com
pki csr test1 CN:example.exampledemo.com C:US L:Purcellville O:Example
OU:Support ST:Virginia subjectAltName:exampledemo.com
pki csr test3 CN:example.exampledemo.com C:US L:Purcellville O:Example
OU:Support ST:Virginia
subjectAltName:exampledemo.com,192.168.1.25,xmpp.exampledemo.com,server.exampledemo.com,join.exampledemo.com,test.exampledemo.com
```

You can include a wildcard for the CN and subjectAltName. For example:
```
pki csr test4 CN:*.exampledemo.com C:US L:Purcellville O:Example
subjectAltName:exampledemo.com
```
Keep the size of certificates and the number of certificates in the chain to a minimum; otherwise TLS handshake round trip times will become long.
### Command/ Examples | Description/Notes
--- | ---
**pki selfsigned** `<key/cert basename>` | Self-signed certificates can be generated for quick testing and debugging. `<key/cert basename>` identifies the key and certificate which will be generated e.g. "pki selfsigned new" creates new.key and new.crt (which is self-signed).

**pki pkcs12-to-ssh** `<username>`

**pki pkcs12-to-ssh john** | Public SSH keys stored in PKCS#12 files can be used but need to be processed first. This command extracts a useable public key from a PKCS#12 file uploaded with the name `<username>.pub`. You are prompted to enter the password for the pkcs#12 file. After completion, the pkcs#12 file is replaced with a useable key without password protection.

Note: Any other data contained in the pkcs#12 file is lost.
The key of an uploaded PKCS#12 file john.pub for user john can be made useable by executing this command.
Cisco Legal Information

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS” WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOSS PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

© 2016–2019 Cisco Systems, Inc. All rights reserved.
Cisco Trademark

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)