Cisco Meeting Server

Cisco Meeting Server Release 3.13

Certificate Guidelines

May 04 2026

Contents

Change History

Date	Change Summary
May 4, 2026	Updated the document for Cisco Meeting Server version 3.13. Chapter 2: Certificates required for the deployment: Added note about TLS Certificate policy update and Cisco Meeting Server 3.13 Compatibility.

1 Introduction

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

Note: Cisco Meeting Server software version 3.0 onwards does not support X-Series servers.

The Cisco Meeting Server is very secure, most of the services and applications running on the server use the TLS cryptyographic 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 ascalable and resilient deployment. Where there may be differences for other deployments, i.e. single split or single combined, these differences are highlighed.

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 2 details where certificates are required within the scalable and resilient server model, and the type of certificates required.

Chapter 3 explains how to create certificates.

Chapter 4 covers installing the certificates on the Meeting Servers.

Chapter 5 provides troubleshooting information for typical certificate related issues.

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

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.

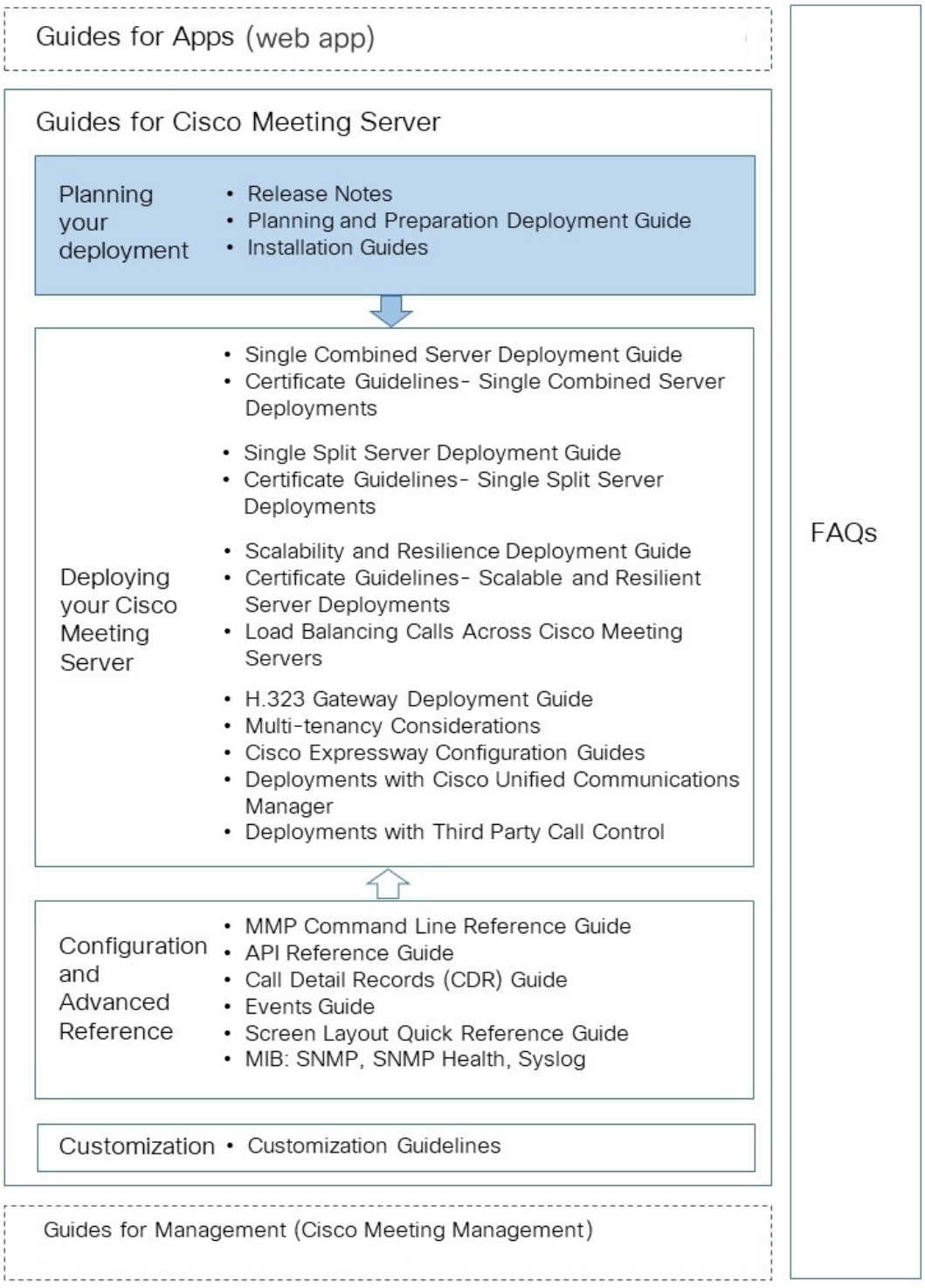
Appendix D provides certificate and configuration information if you are deploying Web Bridge 3 to use Cisco Meeting Server web app.

Note: Cisco Meeting App for WebRTC (Web Bridge 2) is removed from Cisco Meeting Server version 3.0. If using software version 3.0 or later, you will need to use Cisco Meeting Server web app instead of Cisco Meeting App for WebRTC. To do this, you need to deploy Web Bridge 3 – for details on deploying and configuring Web Bridge 3, see the 3.0 or later Deployment Guides.

Important Information: From version 3.0, the XMPP server, Load Balancer, SIP Edge and H.323 Gateway components have been removed from the Cisco Meeting Server software. In addition, new SIP Recorder and Streamer components replace the previous XMPP client versions of the Recorder and Streamer, which have been removed from the server software. The TURN server remains in version 3.0 software and may be used to connect the browser based Cisco Meeting Server web app to Meeting Server conferences. Both native and browser based Cisco Meeting App clients are not supported in version 3.0.

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

Figure 1: Overview of guides covering the 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

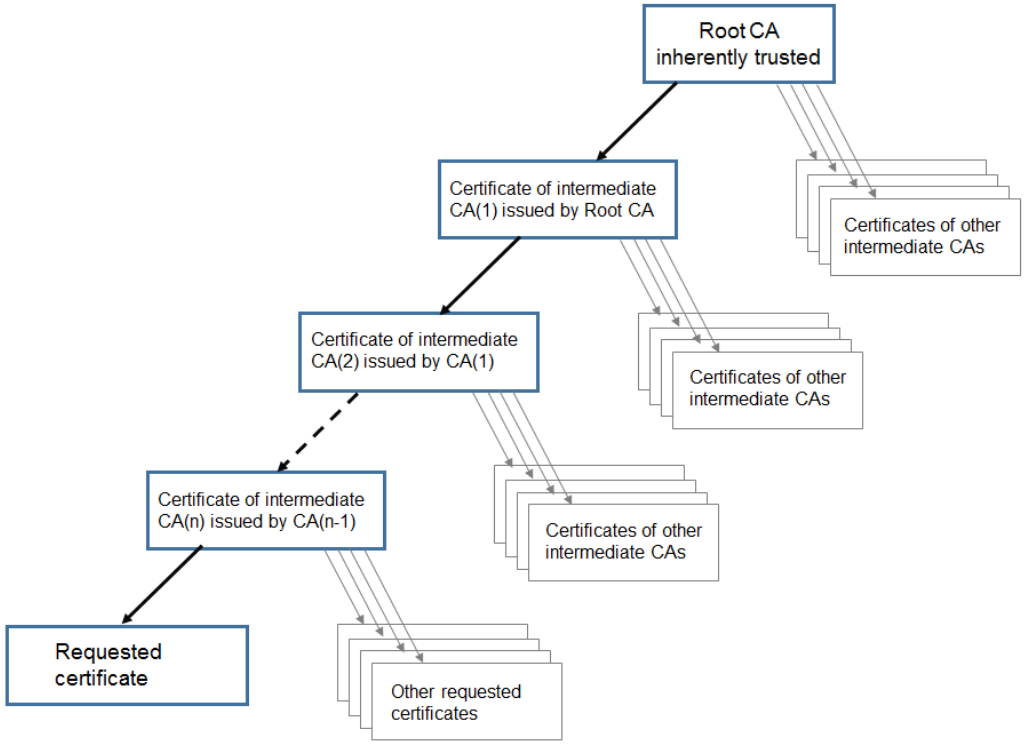
Information	Description
Common Name (CN)	This is the fully qualified domain name that you wish to secure e.g. 'www.example.com'.
Organization or Business name (O)	Usually the legal incorporated name of a company. It should include any suffixes such as Ltd., Inc., or Corp.
Organizational unit or Department name (OU)	For example, Support, IT, Engineering, Finance.

Information	Description
Location (L)	City or town. For example, London, Boston, Milan, Berlin.
Province, Region, County or State (ST)	For example, Buckinghamshire, New Jersey. Do not abbreviate.
Country (C)	The two-letter ISO code for the country where your organization is located. For example, US, GB, FR.
An email address	An email address to contact the organization. Usually the email address of the certificate administrator or IT department.
Subject Alternative Name (subjectAltName)	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.

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.

Figure 2: Certificate Chain of Trust



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) require the certificate and certificate bundle to be presented by the Web Bridge 3 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. Certificate 1 will end with -----END CERTIFICATE----- and the very next line will have -----BEGIN CERTIFICATE---- for 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.

Note: Web Bridge 3 requires all of its certificate definitions to use a bundle of certificates, i.e. a full chain file. This is different to the certificate implementation for Web Bridge 2.

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.

Version 3.0 introduces a change to the C2W connection certificate for Web Bridge 3 so that the trust store no longer needs root certificates as was the case in 2.9. This gives administrators more flexibility on which certificates are trusted. For example, if you need to use a public certificate to protect the C2W connection due to a company's internal policies, now you can

still choose not to trust all the certificates signed by that public CA but trust only the client or server C2W certificate used in the other end. This is called certificate pinning.

2 Certificates required for the deployment

This chapter explains where certificates are required to establish secure connections and the type of certificates required.

Note about components removed from Cisco Meeting Server 3.0 software: The following components were removed from software version 3.0: Web Bridge 2, H.323 Gateway, SIP Edge, XMPP Server and Load Balancer.

Note: Meeting Server 3.13 will no longer enforce client Extended Key Usage (EKU) validation on certificates. All services and components within the Meeting Server will continue to work with the existing certificates without requiring any changes to the certificates in current deployments.
However, for components such as the database that require certificates with the client authentication EKU, it is recommended to generate these certificates through an internal or private Certificate Authority (CA) or use an external certificate authority that issues certificates without the client authentication EKU.

2.1 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 can use certificates signed by a Public or 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.

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 using WebRTC calls, the Call Bridge certificate must include a KeyUsage extension with the digitalSignature bit set, otherwise DTLS negotiation of media between Call Bridge and web app clients may fail. This KeyUsage is normally included when using CA settings for Client and/or Server certificates.

Table 2: Public CA signed certificates (scalable and resilient server model)

Applications requiring public CA signed certificate	Certificate usage	Reason
Web Bridge 3 (only if web app is used)		Web browsers require a public CA signed certificate
Call Bridge (only if Meeting Server connected on a public network for direct Lync federation)	callbridge – TLS Web Server Auth, TLS Web Client Auth	Lync Edge server requires a public CA signed certificate from the Call Bridge if doing direct federation.
TURN server	TURN – TLS Web Server Auth	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

Table 3: Internal CA signed certificates (scalable and resilient server model)

Applications that can use internal CA signed certificate	Certificate usage	Reason
Web Admin	WebAdmin – TLS Web Server Auth	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. In addition, Call Bridges in a cluster connect to each other over HTTPS, via the Web Admin. From version 2.4, you can improve the security of a Call Bridge cluster by using the Call Bridge trust store to validate Call Bridges within the cluster. See Section 4.9.

Applications that can use internal CA signed certificate	Certificate usage	Reason
Call Bridge	TLS Web Client Authentication and TLS Web Server Authentication	The Web Bridge 3 c2w connection 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.
Database client	Databases - TLS Web Client Auth	Database clustering uses public/private key encryption for both confidentiality and authentication. Each server hosting a database requires a set of certificates signed by the same CA. See Section 3.1.4.
Database server	Databases - TLS Web Server Auth	
Recorder	Recorder - TLS Web Server Auth	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.
Streamer	Streamer - TLS Web Server Auth	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.
c2w	webbridge 3 c2w - TLS Web Server Auth	C2W certificates are used for the connection between Call Bridge and Web Bridge 3.

Note: Certificate usage is noted to help specify certificate properties when a certificate is signed by the CA. If ExtendedKeyUsages are not enabled, the certificate is valid for all uses. If any are enabled, the certificate's ExtendedKeyUsages must include at least the usages defined in the table.

Note: The KeyUsage bits for the Call Bridge certificate must include KeyUsage digitalSignature or DTLS negotiation of media between Call Bridge and web app clients may fail. This KeyUsage is normally included when using CA settings for Client and/or Server certificates.

2.2 How Meeting Server validates certificate extensions

Meeting Server validates certificate extensions as detailed below.

For client connections:

- If an ExtendedKeyUsage extension is present, it must have the TLS Web Client Authentication bit set
- If a KeyUsage extension is present, it must have at least one of the digitalSignature and keyAgreement bits set
- If a Netscape extension is present, it must have the SSL Client bit set

For server connections:

- If an ExtendedKeyUsage extension is present, it must have the TLS Web Server Authentication bit set
- If a KeyUsage extension is present, it must have at least one of the digitalSignature, keyEncipherment and keyAgreement bits set
- If a Netscape extension is present, it must have the SSL Server bit set

Additionally, when validating a chain of certificates, all certificates except the leaf certificate must be marked as CA with at least one of the following options:

- If a KeyUsage extension is present, it must have the keyCertSign bit set
- If a basic constraints extension is present, it must have the CA bit set
- If a Netscape extension is present, it must have the SSL CA bit set

3 Obtaining certificates

Section 2.1 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. Use SFTP to upload to the Meeting Server the signed certificate and intermediate bundle (if any) from the CA.

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 Section 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 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 1 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

Attribute		Description	Optional/Required
CN	Common Name	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.	Required, see notes below
O	Organization or Business name	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."	Optional
OU	Organizational unit or Department name	For example, Support, IT, Engineering, Finance. Use "" around the attribute if more than one word, e.g. "Human Resources"	Optional
L	Location	City or town. For example, London, Boston, Milan, Berlin.	>Optional
ST	Province, Region, County or State	For example, Buckinghamshire, California. Do not abbreviate. Use "" around the attribute if more than one word, e.g. "New Jersey"	Optional

Attribute		Description	Optional/Required
C	Country	The two-letter ISO code for the country where your organization is located. For example, US, GB, FR.	Optional
An email address		An email address to contact the organization. Usually the email address of the certificate administrator or IT department.	Optional
SAN	Subject Alternative Name	From X509 Version 3 (RFC 2459), SSL certificates are allowed to specify multiple names that the certificate should match. This optional 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 **CN** in this list.	Required or if a single certificate is to be used across multiple components. See note below

Points to note:

- If you plan to use a dedicated certificate for the Web Bridge 3 then specify in the CN field, the FQDN that is defined in the DNS A record for the Web Bridge3. Failure to specify the FQDN may result in browser certificate errors.
- If you plan to use the same certificate across multiple components, for example the Web Bridge 3, 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=**callbridge1.example.com,callbridge2.example.com,callbridge3.example.com, webbridge3.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:callbridge1.example.com,callbridge2.example.com,callbridge3.example.com,webbridge3.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 each Call Bridge is used within your deployment, it may need private key/ certificate pairs:

- to establish communication with the Web Bridge 3. It is important for the security of the deployment that configuration is only accepted from Call Bridges that are trusted.
- 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:
  - For single split and single combined deployments: 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.
  - For scalable and resilient deployments: the **CN** in the Certificate must be the same as the FQDN that was added when configuring the Meeting Server as a trusted application on the Lync FE server. To see this value, you can use the Lync Powershell command **get-cstrustedapplicationcomputer**.
  - 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 4.4 provides details on uploading certificates for the Call Bridges.

3.1.2 CSR for the Web Bridge 3

Web browsers look at the **CN** field to determine the Web Bridge 3's FQDN. To avoid web browser certificate errors, follow this advice:

- if you are using a dedicated certificate for the Web Bridge 3: in the **CN** field, specify the FQDN that is defined in the DNS A record for the Web Bridge 3. 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 append the **CN** to the **SAN** list if a **SAN** list exists.

Note: If your deployment requires the Cisco Expressway Web Proxy to connect to the Web Bridge, then ensure the SAN field of the Web Bridge certificate includes the A record used by the Expressway-C that will connect to the Web Bridge, otherwise the connection will fail. For example, if the Expressway is configured to connect to the Web Bridge on join.example.com, an A record must exist for this FQDN, and the SAN field of the Web Bridge certificate must include join.example.com.

- if you plan to use the same certificate across multiple components (Web Bridge 3, 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 webbridge3 CN:www.example.com O:"Example Inc."
```

or

```
pki csr webbridge3 CN:www.example.com O:"Example Inc."
subjectAltName:guest.example.com
```

The example will generate two files: webbridge3.key and webbridge3.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 4.5 provides details on uploading certificates for the Web Bridge 3s.

3.1.3 CSR for the MeetingApps

Web browsers communicate directly with MeetingApps, hence it required to use a browser trusted CA certificates. If you plan to use an internal CA signed certificate, they must be validated on each browser you use for web app.

Note: If you are using internal CA signed certificates, file share feature might not work on all browsers and all operating systems.

It is recommended use a dedicated certificate for the MeetingApps. Web browsers look at the **CN** field to determine the MeetingApps FQDN. In the **CN** field, specify the FQDN that is defined in the DNS A record for the MeetingApps. 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 append the **CN** to the **SAN** list if a **SAN** list exists.

For example:

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

or

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

The example will generate two files: meetingapps.key and meetinapps.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 4provides details on uploading certificates for MeetingApps

3.1.4 CSR for database clustering

Note: This section only applies to scalable and resilient deployments.

From version 2.7, database clusters require client and server certificates signed by the same CA configured in each Meeting Server holding or connecting to a database in the cluster. Enforcing the use of certificates ensures both confidentiality and authentication across the cluster.

From 2.7 you can generate all certificates directly from the MMP. Certificates and keys then need to be downloaded via SFTP and uploaded to each Meeting Server that belongs to the same database cluster.

CAUTION: The nodes forming a database cluster must be configured with a trusted root CA certificate so that only legitimate nodes can connect to the cluster. The nodes will trust connections that present a certificate chain that ends with a trusted root certificate. Therefore each database cluster must use a dedicated root certificate, the root certificate or intermediate certificates must not be used for any other purpose.

You can still use other methods to create certificates, for example openssl. See for more information.

For the database clustering:

1. Create the dbca selfsigned certificate with the pki selfsigned command.

   For example:

   ```
   pki selfsigned dbca CN:"My company CA"
   ```

   creates a local private key named **dbca.key** and a self-signed certificate with the common name **CN=My company CA** in dbca.crt

2. Create a private key and Certificate Request File for the database server. You can use the same certificate on all of the servers in the database cluster; specify the FQDN of one of the

servers in the CN field and specify the FQDN of the other servers in the SAN field. If using "Extended Key Usage", ensure "Server Authentication" is allowed for the database server.

For example:

```
pki csr dbserver CN:server.db.example.com
subjectAltName:server02.db.example.com
```

generates a CSR file named dbserver.csr and private key named dbserver.key.

3. Create a private key and Certificate Request File for the database client. The CommonName (CN) for a database client must equal 'postgres'. If using "Extended Key Usage", ensure "Client Authentication" is allowed for the database client.

   For example:

   ```
   pki csr dbclient CN:postgres
   ```

   generates a CSR file named dbclient.csr and private key named dbclient.key

4. Use an Internal CA to sign the dbserver.csr and dbclient.csr certificate signing request files and obtain the corresponding dbserver.crt and dbclient.crt certificates, as well as the Internal CA certificate (bundle).

   For example:

   ```
   pki sign dbserver dbca
   pki sign dbclient dbca
   ```

1. Create a private key and Certificate Request File for the database server. You can use the same certificate on all of the servers in the database cluster; specify the FQDN of one of the servers in the CN field and specify the FQDN of the other servers in the SAN field. If using "Extended Key Usage", ensure "Server Authentication" is allowed for the database server.

   For example:

   ```
   pki csr db01server CN:www.example.com
   ```

   generates a CSR file named db01server.csr and private key named db01server.key.

2. Create a private key and Certificate Request File for the database client. The CommonName (CN) for a database client must equal 'postgres'. If using "Extended Key Usage", ensure "Client Authentication" is allowed for the database client.

   For example:

   ```
   pki csr db01client CN:postgres
   ```

   generates a CSR file named db01client.csr and private key named db01client.key

3. Use an Internal CA to sign the db01server.csr and db01client.csr certificate signing request files and obtain the corresponding db01server.crt and db01client.crt certificates, as well as the Internal CA certificate (bundle). See Section 3.3

Section 4.4 provides details on uploading certificates for database clustering.

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 3, so that WebRTC clients trusts the connection. The certificate should be signed by the same Certificate Authority as used for the Web Bridge 3 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 4.6 provides details on uploading certificates for the TURN servers.

3.2 Signing the CSR using a public Certificate Authority

Refer to Section 2.1 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 for a list of public CA signed certificates required for the Meeting Server and their usage requirements. 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: We recommend that you verify that all your certificates have the correct CN, SAN, KeyUsage, and ExtendedKeyUsage values using the `pki inspect` command or openssl tools before deploying the certificates.

3.3 Signing the CSR using an internal Certificate Authority

Refer to Section 2.1for a list of Internal CA signed certificates required for the Meeting Server and their usage requirements.

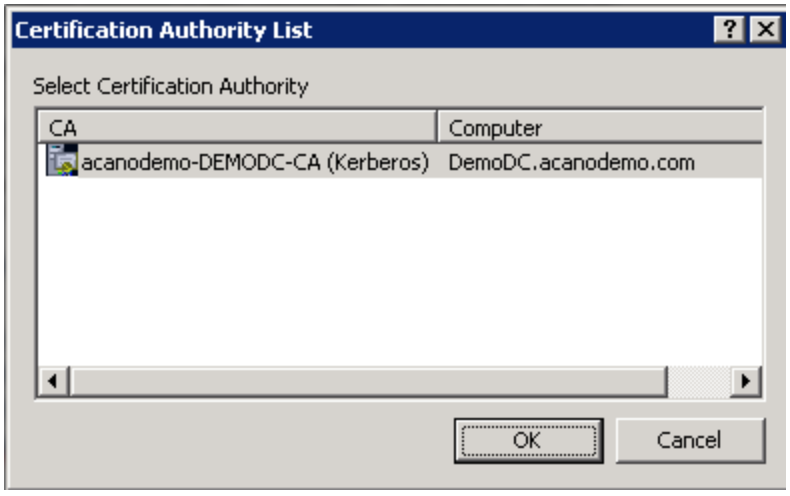
Note: In deployments where the Meeting Server is trunked to Cisco Unified Communications Manager, Cisco Unified Communications Manager requires the Call Bridge certificate to be signed using a template that allows for an Extended Key Usage containing both TLS Web Client Authentication and TLS Web Server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

Note: We recommend that you verify that all your certificates have the correct CN, SAN, KeyUsage, and ExtendedKeyUsage values using the **`pki inspect`** command or openssl tools before deploying the certificates.

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:Computer" <path\csrfilename>
   ```
   For example:
   ```
   certreq -submit -attrib "CertificateTemplate:Computer"
   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.

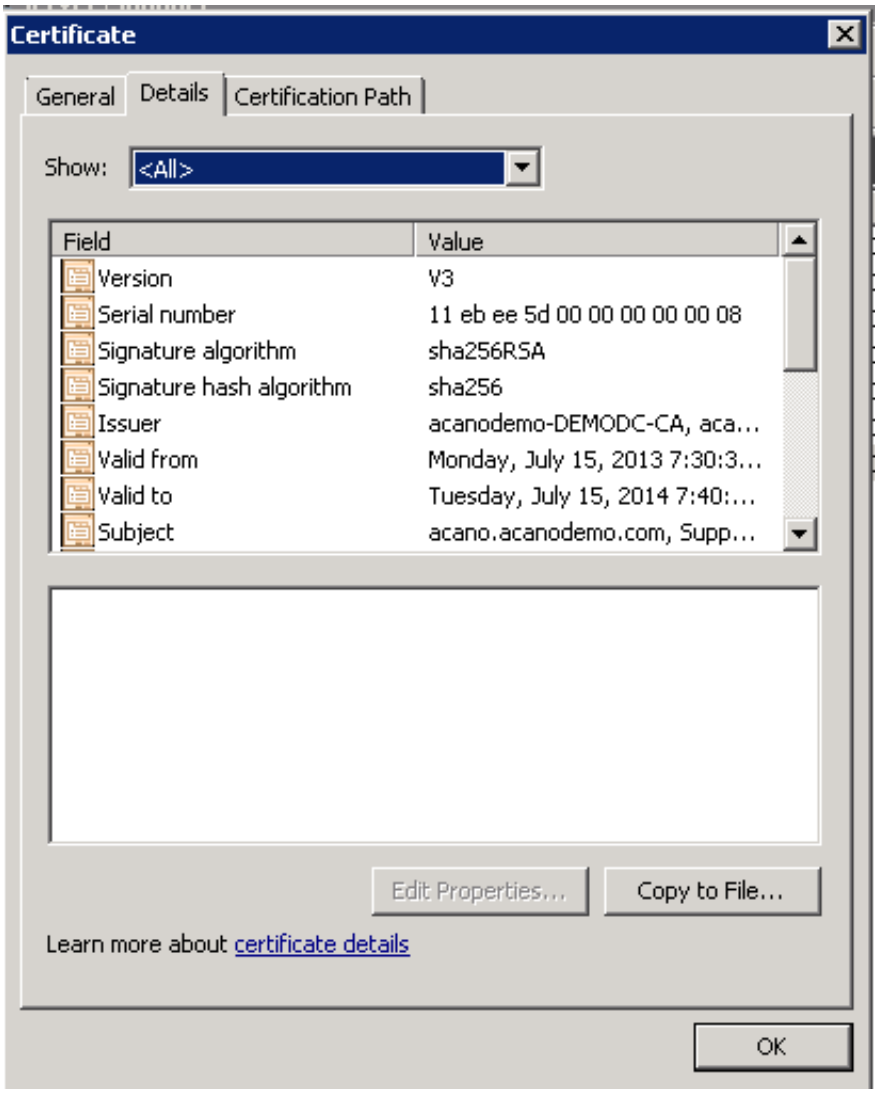


   If your Windows account has permission to issue certificates, you will be prompted to save the resulting certificate. Save the file with a .crt, .cer or .pem extension, for example example.crt. Go to step 4. See Appendix B for a brief overview of certificate file extensions.

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.

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 **`callbridge`** and click **Next**.

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

The scalable and resilient Meeting Server deployment requires public CA signed certificates for:

- the **TURN server**, if you plan to use TLS connections for secure communication.
- 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.
- the **Web Bridge 3**, if you wish to deploy Web Bridge 3 to use web app, it requires a public CA signed certificate from the Call Bridge in order to trust the connection.

And Public or 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 3 c2w connection requires and needs to trust 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.
- **Database** clustering where each database server and database client (including Call Bridges not co-located with a database) requires a private key and certificate signed by the same Certificate Authority. (For scalable and resilient deployments only.)
- 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 Scalable and Resilient 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 Call Bridge. Refer to the Streamer section in the Cisco Meeting Server Scalable and Resilient deployment guide for details on uploading the certificates and configuring the Streamer.

- If you enable **Uploader**, it requires a certificate bundle containing the Root CA and intermediate certificates for both the Vbrick Rev Server and the Cisco Meeting Server's Web Admin interface. This bundle must be trusted by the Cisco Meeting Server to enable secure HTTPS communication for uploading recordings to Vbrick Rev without trust issues. Refer to the Configuring the Meeting Server to work with Vbrick section in the Cisco Meeting Server Single Split Server deployment guide.
- If you enable **Web Bridge 3**, C2W certificates are used for the connection between Call Bridge and Web Bridge 3. For the Call Bridge to make a C2W connection to a Web Bridge 3, you need to specify a C2W trust store to verify certificates against.
- The certificates used for the **MeetingApps** must be assigned while configuring the MeetingApps using the MMP commands. If you are using internal CA signed certificates for MeetingApps, they must be validated by connecting to the MeetingApps address on every browser.

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: **`webadmin.crt`** and **`webadmin.key`**.

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 callbridge.crt`

   to inspect the contents of the file, for instance to see whether a certificate is still valid.
3. Enter the command:

   `pki match callbridge.key callbridge.crt`

   to check that the file **`callbridge.key`** matches file **`callbridge.crt`** and together they form one usable identity.
4. Enter the command:

   `pki verify callbridge.crt callbridgebundle.crt`

   to check that callbridge.crt is signed by a trusted CA, with the chain of trust established through the chain of intermediate certificates in callbridge.crt.

4.4 Installing the Certificate and Private Key for the Call Bridges

As explained in Section 1.1.1, depending on how each 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 each Call Bridge will use to listen. Refer to the Scalable and Resilient 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
```

Note: You will need to add the Call Bridge certificate to every Web Bridge 3's trust store after you've configured the Web Bridge 3s.

Note: Use the MMP command **callbridge certs none** to remove the certificate configuration from the Call Bridge.

4.4.1 Establishing Trust between the Call Bridge and the Web Bridge 3

The Web Bridge 3 allows configuration of guest logins and image customizations to be pushed from a Call Bridge (see the Customization Guidelines). It is important for the security of the deployment that configuration is only accepted from Call Bridges that are trusted.

C2W certificates are used for the connection between Call Bridge and Web Bridge 3. For the Call Bridge to make a C2W connection to a Web Bridge 3, you need to specify a C2W trust store to verify certificates against. For more information, see Section 4.5 and Appendix D.

4.5 Installing the Certificates and Private Keys for Web Bridge 3

For full details on certificate configuration to deploy Web Bridge 3 to use Cisco Meeting Server web app, see Appendix D. The following is overview information to help you configure Web Bridge 3 so that you can use web app.

- Web Bridge 3 requires all of its certificate definitions to use a bundle of certificates, i.e. a full chain file. This is different to how certificates are implemented for Web Bridge 2.
- " Call Bridge to Web Bridge" protocol (C2W) is the link between the Call Bridge and Web Bridge 3.

- A port must be opened on an interface (using **`webbridge3 c2w listen`**) to allow the Call Bridge to connect to the Web Bridge 3 (the Web Bridge 3 listens on that port). This is why you have to give the address with this port when you do the API request to tell a Call Bridge about this Web Bridge 3. This connection must be secured with certificates.
- We recommend you protect that opened port from external access – it only needs to be reachable from Call Bridges.
- The Call Bridge uses the certificate set using **`callbridge certs`** and the Web Bridge 3 uses the certificate set using **`webbridge3 c2w certs`**.
- The Web Bridge 3 will trust certificates of Call Bridges that have been signed by one of those in its trust store, set by **`webbridge3 c2w trust`**.
- The Call Bridge will trust Web Bridge 3s that have certificates signed by one of those in its trust store, set by **`callbridge trust c2w`**.
- You do not need a certificate signed by a public authority – you can use self-signed certificates created within the MMP.
- The SAN/CN must match the FQDN or IP address that is used in the c2w:// url used to register the Web Bridge 3 in the Call Bridge API. (If this does not match, the Call Bridge will fail the TLS negotiation, rejecting the certificate presented by the Web Bridge 3, and will fail to connect with the Web Bridge 3.)
- For details on Call Bridge and Web Bridge 3 certificate extension requirements, see Chapter 2.1.

4.6 Installing the Certificates and Private Keys for TURN Servers

If you plan to use TLS for secure communication, then you need to install a signed certificate for the TURN servers, 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 Scalable and Resilient 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. Use SFTP to upload to the Meeting Server the signed certificate and intermediate bundle (if any) from the CA.

4. Check that the certificate (and certificate bundle) and the private key match

   ```
   pki verify <certicate> <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
   ```

4.7 Installing the Certificates for MeetingApps

Before the participants can share files in a web app meeting, the certificates for the MeetingApps must be configured. It is recommended to use publicly signed browser trusted CA certificates. However, if you are using internal CA signed certificates, web app prompts you to validate these certificates on each browser that you use for joining meetings.

You must configure the interface and the port that the MeetingApps will use to communicate before assigning the certificates. Refer to *Cisco Meeting Server Deployment guide* for details on configuring MeetingApps interface and port.

To configure MeetingApps certificates:

1. SSH into the MMP of the host server.
2. Disable MeetingApps before assigning the certificate

   ```
   meetingapps disable
   ```

3. Assign the certificate key pair for the MeetingApps using the command

   ```
   meetingapps https certs <key-file> <crt-fullchain-file>
   ```

4. Enable MeetingApps

   ```
   meetingapps enable
   ```

4.7.1 Validating MeetingApps certificate

In order to trust the connection, the internal CA signed certificates used for the MeetingApps must be validated on each browser that you use for web app meetings. Participants cannot share the files in meeting if the internal CA signed certificates are not validated and web app prompts the participants with a link that can be used validate the MeetingApps certificates.

To validate the certificate, open a new tab on the browser and enter the MeetingApps address followed by **:** and the port number. When prompted, you should accept the certificate and

continue to access the MeetingApps. The address and the port used for the MeetingApps can be retrieved using the **meetingapps** command.

Note: You must repeat the process on every browser that you use for web app meetings.

You can share the files in a web app meeting after you have validated the MeetingApps certificates.

4.7.2 Adding certificates to Trusted Root Certificate Authorities

Alternatively, you can also install the internal CA signed certificate into the Trusted Root Certification Authorities store of every browser that you use for web app meetings.

To add the internal CA signed certificate into the Trusted Root Certification Authorities store:

1. Download the intermediate and the root certificates used by the MeetingApps.
2. Import all the internal CA signed certificates to Trusted Root Certification Authorities.

Once you have imported the certificates to the trust store, you can join the web app meeting and share files.

4.8 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.

4.9 Call Bridge cluster validation

Note: This section only applies to scalable and resilient deployments.

You can improve the security of a Call Bridge cluster by using the Call Bridge trust store to validate Call Bridges within the cluster. As Call Bridges connect to each other over HTTPS, which is fronted by the Web Admin, you need to create a certificate bundle holding the Web Admin certificates of the clustered Call Bridges, and upload the certificate bundle to the trust store of each Call Bridge in the cluster. Use the MMP command:

```
callbridge trust cluster <bundle name>
```

When a Call Bridge connects to another Call Bridge in a cluster, it checks the bundle of certificates in its trust store to validate the identity of the Call Bridge that it is connecting to. This removes the risk that the Call Bridge is connecting to an insecure Meeting Server. The certificate bundle can be either a certificate chain or an allowed list of trusted certificates.

Note: From version 3.4, certificate name validation has been implemented when '**`callbridge trust cluster`**' is enabled and thus the peers configured on the clustering must match the exact FQDN on its corresponding Web Admin certificate and failure to make this configuration will result in Call Bridge cluster failure.

If the trust store is not used, then there will be no certificate validation between clustered Call Bridges, and a Call Bridge will continue to make the connection to a remote Call Bridge, but without verifying its identity.

To remove the Call Bridge cluster certificate bundle from the Call Bridge trust store, use the MMP command:

```
callbridge trust cluster none
```

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 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.

5.3 Certificate soon to expire or already expired message

Renewing a certificate follows the same process as deploying a new set of certificates. Refer back to the earlier sections on obtaining and installing certificates.

In case of a clustered environment, once the certificate expires, database nodes in cluster stops talking to each other. Certificates cannot be renewed on Meeting Server database cluster nodes unless cluster is removed using the command **`database cluster remove`**. Hence uncluster the database, update the certificates and create the cluster again. For details refer How to renew expired database cluster certificates using Cisco Meeting Server.

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 'cluster keys' (as no CA certificate can be obtained (for scalable and resilient deployments only) or Lync authentication (as CA is not a trusted authority). However, self-signed certificates can be used for Web Admin, 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 Section 1.2, 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.

Note: The examples in this chapter use Web Bridge3.

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 webbridge3.csr -new -newkey rsa:2048 -nodes -keyout
webbridge3.key
```

generates a CSR file named webbridge3.csr and an RSA 2048 bit private key named webbridge3.key,

Note: The keyname and certname must NOT include a "." or "_", for example **webbridge3** is valid, but **web.bridge 3** or **web_bridge_3** 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 webbridge3.csr -key webbridge3.key –new
```

generates a CSR file named webbridge3.csr based on an existing private key named webbridge3.key,

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

A.1.1 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.2 Generating ECDSA private keys and CSR files

A.2.1 Generating Root CA Certificate for signing certificate request

To generate the root private key, use the command:

```
openssl ecparam -genkey -name prime256v1 -out root.key
```

This command generates a private key named " root.key" using the elliptic curve prime256v1.

To Generate the ECDSA Root CA Certificate

```
openssl req -new -x509 -SHA256 -key root.key -out root.crt -days 3650
-
subj"/C=<country>/ST=<state>/L=<location>/O=<organization>/OU=<organiz
ational unit>/CN=<authorityname>"
```

For example:

```
openssl req -new -x509 -SHA256 -key root.key -out root.crt -days 3650 -
subj"/C=UK/ST=London/L=London/O=Example/OU=/CN=example"
```

This command generates a self-signed certificate named " root.crt" valid for 3650 days (approximately 10 years).

A.2.2 Generating Intermediate CA Certificate for signing certificate request

To generate the Intermediate certificate private key, use the command:

```
openssl ecparam -genkey -name prime256v1 -out intermediate.key
```

This command creates a private key named " intermediate.key" using the elliptic curve prime256v1.

To generate a Server certificate signing request (CSR)

1. Create an extension file (.conf) with the following attributes:

   [req]
   distinguished_name = req_distinguished_name
   req_extensions = v3_req
   prompt = no

[req_distinguished_name]
C = <country>

ST = <state>

L = <location>

O = <organization>

OU = <organization unit>

CN = <commonname/hostname>

[v3_req]

basicConstraints = CA:TRUE, pathlen:0

2. Use the command:
   openssl req -new -sha256 -key intermediate.key -out intermediate.csr <.conf file>
   This command creates a CSR named " intermediate.csr" using the Intermediate CA private key.

A.2.3 Generating Server Certificate for signing certificate request

To generate the server certificate private key, use the command:

openssl ecparam -genkey -name prime256v1 -out server.key

This command creates a private key named " server.key" using the elliptic curve prime256v1.

To generate a Server certificate signing request (CSR)

1. Create an extension file (.conf) with the following attributes:

   [req]
   distinguished_name = req_distinguished_name
   req_extensions = v3_req
   prompt = no

   [req_distinguished_name]
   C = <country>

   ST = <state>

   L = <location>

   O = <organization>

   OU = <organization unit>

   CN = <commonname/hostname>

[v3_req]

keyUsage=digitalSignature, keyEncipherment, dataEncipherment

extendedKeyUsage=serverAuth, clientAuth

subjectAltName=@alt_names

[alt_names]

DNS.1 = <DNS hostname>

2. Use the command:
   openssl req -new -SHA256 -key server.key -out server.csr -config <.conf file> Use the .conf file created in step 1.
   This command generates a CSR named " server.csr" using the server private key.

To sign the Server Certificate with the Intermediate CA

```
openssl x509 -req -in server.csr -CA intermediate.crt -CAkey
intermediate.key -CAcreateserial -out server.crt -days 365 -SHA256 -
extfile <.conf file> -extensions v3_req
```

This command signs the server CSR with the Intermediate CA certificate and private key. It generates a certificate named " server.crt" valid for 365 days. The extensions specified in the " extensions.txt" file are included in the server certificate.

To self-sign the certificate, use the command

```
openssl x509 -req -in server.csr -CA root.crt -CAkey root.key -
CAcreateserial -out server.crt -days 365 -SHA256 -extfile <.conf file>
-extensions v3_req
```

This command signs the server CSR with the root CA certificate and private key. It generates a certificate named " server.crt" valid for 365 days. The extensions specified in the " extensions.txt" file are included in the server certificate.

A.3 Creating Certificates for Database Clustering

This section is for scalable and resilient deployments only and details how to create certificates for database clustering using OpenSSL commands.

Note: As database cluster certificates are mandatory from 2.7, to make it easier to setup Meeting Server database clustering you can use pki commands to create signed certificates for the database cluster. For more information, see Section 3.1.4.

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.

Database clusters require client and server certificates signed by the same CA configured in each Meeting Server holding or connecting to a database in the cluster. Enforcing the use of certificates ensures both confidentiality and authentication across the cluster.

CAUTION: If a database cluster was configured without certificates using an earlier version of Meeting Server software which did not require certificates, then on upgrading to version 2.7 the database will stop and remain unreachable until certificates are configured and the database cluster is recreated.

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/OU=/CN=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 db01ca.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/OU=/CN=server1
```

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 1.8 to upload and assign the database certificates and private keys .

a. Each server hosting a database, requires 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 Creating ECDSA certificates for Database clustering

To generate ECDSA certificate for a clustered database, follow the steps mentioned in Section A.2. However, the DNS must be configured as explained below:

For Database servers, the DNS must be configured to **`nodenames`** of all the servers hosting the database.

For Database clients, the DNS must be configured to " postgres" .

A.5 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

Extension	Information on file type
.pem	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.
.der	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.
.cer	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.
.crt	Certificate used by secure websites (beginning with "https://") 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.

Table 6: Permitted extensions for private key files

Extension	Information on file type
.key	Used both for public and private PKCS#8 keys. The keys may be encoded as binary DER or as ASCII PEM.
.pem	Indicates key was encoded using PEM (ASCII base64).
.der	Indicates key was encoded using binary DER.

Appendix C MMP PKI commands

Below is a list of MMP pki commands:

Command/Examples	Description/Notes
pki	Displays current PKI usage.
pki list	Lists PKI files i.e. private keys, certificates and certificate signing requests (CSRs).
pki inspect <filename>	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.
pki match <key> <certificate>	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. callbridge.
pki verify <cert> <cert bundle/CA cert> [<CA cert>] **pki verify server.pem bundle.pem rootca.pem** **pki verify server.pem bundle.pem**	A certificate may signed by a certificate authority (CA) and the CA will provide a " certificate bundle" of intermediate CA certificates and perhaps a CA certificate in its own file. To check that the certificate is signed by the CA and that the certificate bundle can be used to assert this, use this command.
pki unlock <key>	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

Command/Examples	Description/Notes
pki csr <key/cert basename> [<attribute>:<value>] **pki csr example CN:www.example.com OU:"My Desk" O:"My Office" L:"San Jose" ST:California C:US**	For users happy to trust that Cisco meets requirements for generation of private key material, private keys and associated Certificate Signing Requests can be generated. <key/cert basename> is a string identifying the new key and CSR (e.g. " new" results in " new.key" and " new.csr" files) Attributes for the CSR can be specified in pairs with the attribute name and value separated by a colon (" :"). 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 emailAddress: email address The CSR file can be downloaded by SFTP and given to a certificate authority (CA) to be signed. (Alternatively, the CSR file can be used in the 'pki sign' command to generate a certificate locally.) On return it must be uploaded via SFTP. It can then be used as a certificate. Note: Since 1.6.11 **pki csr <key/cert basename> [<attribute>:<value>]** now takes 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 test2 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,exampledemo.com, server.exampledemo.com,join.exampledemo.com, test.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> [<attribute>:<value>]	You can use this command to generate self-signed certificates. <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). Attributes for the CSR can be specified in pairs with the attribute name and value separated by a colon (" :"). 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 emailAddress: email address 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. 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.
pki sign <csr/cert basename> <CA key/cert basename>	This command signs the csr identified by <csr/cert basename> and generates a certificate with the same basename, signed with the CA certificate and key identified by <CA key/cert basename>. The files <csr/cert basename> and <CA key/cert basename> should have been generated by the commands 'pki csr' and 'pki selfsigned' respectively.
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

Appendix D Certificate and configuration information to deploy Web Bridge 3 to use Cisco Meeting Server web app

D.1 Configuring Meeting Server to use Web Bridge 3

Web Bridge 3 requires configuring an HTTPS port and a C2W port.

To configure Meeting Server to use Web Bridge3:

1. SSH into the MMP and log in.
2. Use the **webbridge3** command in the MMP to configure webbridge3. To display the webbridge 3 usage, enter: **help webbridge3**

```
> help webbridge3

    Usage:
    webbridge3
    webbridge3 restart
    webbridge3 enable
    webbridge3 disable
    webbridge3 https listen <interface:port allowed list>
    webbridge3 https certs  <key-file> <crt-fullchain-file>
    webbridge3 https certs none
    webbridge3 http-redirect (enable [port]|disable)
    webbridge3 c2w listen <interface:port allowed list>
    webbridge3 c2w certs  <key-file> <crt-fullchain-file>
    webbridge3 c2w certs none
    webbridge3 c2w trust <crt-bundle>
    webbridge3 c2w trust none
    webbridge3 options <space-separated options>
    webbridge3 options none
    webbridge3 status
```

3. (Optional) Set up a port for HTTP connections. This port will be opened for all Meeting Server interfaces on which the web app has been configured. Incoming HTTP connections will be automatically redirected to the matching HTTPS port for the interface they arrived on. The default port, if you don't specify one in **webbridge3 http-redirect enable [port]**, is 80.

4. Configure the port for the HTTPS service to listen to. To configure it to listen on port 443 of the a interface:

   ```
   webbridge3 https listen a:443
   ```

5. Set the HTTPS certificates. These are the certificates that will be presented to web browsers so they need to be signed by a certification authority and the hostname/purpose etc needs to match. (The certificate file is the full chain of certificates that starts with the end entity certificate and finishes with the root certificate.) Enter the command:

   ```
   webbridge3 https certs wb3-https.key wb3-https-fullchain.crt
   ```

6. Configure the C2W connection. We recommend that you make this address/port accessible from the Call Bridge(s) only. The following command sets it in port 9999 of interface a:

   ```
   webbridge3 c2w listen a:9999
   ```

   Note that here we use the example of port 9999, however, it can be any available port on your network. It's not a fixed port, unlike 443.

7. Configure the C2W connection certificates. You need to configure the SSL Server certificates used for the C2W connection. (See " Configuring Call bridge to use C2W connections" below for certificate requirements, and more information can be found in this FAQ.)

   ```
   webbridge3 c2w certs wb3-c2w.key wb3-c2w-fullchain.crt
   ```

8. The Web Bridge 3 C2W server is expecting Call Bridges to present a client certificate — it will verify whether to trust them using the trust bundle provided by the following command:

   ```
   webbridge3 c2w trust wb3-c2w-trust-bundle.crt
   ```

9. Now enable Web Bridge 3:

   ```
   webbridge3 enable
   ```

D.2 Configuring Call bridge to use C2W connections

C2W certificates are used for the connection between Call Bridge and Web Bridge 3. For the Call Bridge to make a C2W connection to a Web Bridge 3, you need to specify a C2W trust store to verify certificates against, i.e. the ones presented by the Web Bridge 3 that were configured in step 7 above.

1. Use the **`callbridge`** command in the MMP to display the Call Bridge usage, enter: **`help callbridge`** to display:

```
> help callbridge
Configure CMS callbridge

Usage:

    callbridge listen <interface allowed list>
    callbridge prefer <interface>
    callbridge certs <key-file> <crt-file> [<cert-bundle>]
    callbridge certs none
    callbridge trust c2w <bundle>
    callbridge trust c2w none
    callbridge add edge <ip address>:<port>
    callbridge del edge
    callbridge trust edge <trusted edge certificate bundle>
    callbridge trust cluster none
    callbridge trust cluster <trusted cluster certificate bundle>
    callbridge restart
```

2. Set the certificates for the Call Bridge:

   ```
   callbridge certs cert.key cert.crt
   ```

3. Set the C2W trust store that will be used to validate the SSL Server certificate presented by the Web Bridge 3. (For more information, see this FAQ.)

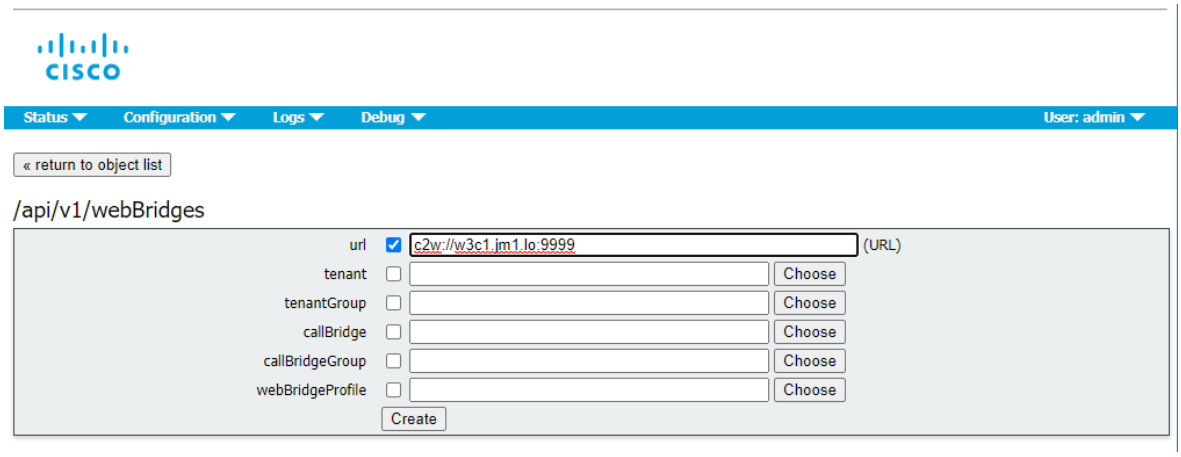
   ```
   callbridge trust c2w c2w-callbrige-trust-store.crt
   ```

4. Now restart Call Bridge:

   ```
   callbridge restart
   ```

5. Go to the Web Admin user interface, select **Configuration > API** and select **/api/v1/webBridges** to register the Web Bridge 3 URL to the running callbridge REST API as shown below. The URL protocol indicates it is webbridge3, i.e. specify c2w:// protocol in the URL so it is handled as a webbridge3 connection.

Figure 4: Registering Web Bridge 3 URL to the Call Bridge API



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, LOST 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.

Cisco Trademark