Generate CSR and Apply Certificates to CMS

Contents

Introduction

This document describes how to generate Certificate Signing Request (CSR) and upload signed certificates to Cisco Meeting Server (CMS).

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Basic knowledge of CMS Server

Components Used

The information in this document is based on these software and hardware versions:

- Putty or similar software
- CMS 2.9 or later

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Generate the CSR

There are two ways you can generate CSR, one of those is to generate the CSR directly on the CMS server from Command Line Interface (CLI) with admin access, the other is to do it with external 3rd party

Certificate Authority (CA) such as Open SSL.

In both cases, the CSR has to be generated with the correct syntax for CMS services to work properly.

Step 1. Syntax Structure

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

- **<key/cert basename>** is a string that identifies the new key and CSR name. It can contain alphanumeric, hyphen or underscore characters. This is a mandatory field.
- **<CN:value>** is the Common Name. This is the Fully Qualified Domain Name (FQDN) that specifies the server's exact location in the Domain Name System (DNS). This is a mandatory field.
- **[OU:<value>]** is the Organizational Unit or Department Name. For example, Support, IT, Engineer, Finance. This is an optional field.
- **[O:<value>]** is the Organization or Business name. Usually the legally incorporated name of a company. This is an optional field.
- **[ST:<value>]** is the Province, Region, County or State. For example, Buckinghamshire California. This is an optional field.
- **[C:<value>]** is the Country. The two-letter International Organization for Standardization (ISO) code for the country where your organization is located. For example, US, GB, FR. This is an optional field.
- **[subjectAltName:<value>]** is the Subject Alternative Name (SAN). From X509 Version 3 (RFC 2459), Secure Socket Layers (SSL) certificates are allowed to specify multiple names that the certificate must match. This field enables the generated certificate to cover multiple domains. It can contain IP addresses, domain names, email addresses, regular DNS hostnames, etc, separated by commas. If it is specified, you must also include the CN in this list. Although this is an optional field, the SAN field must be completed in order for Extensible Messaging and Presence Protocol (XMPP) clients to accept a certificate, otherwise the XMPP clients display a certificate error.

Step 2. Generate Callbridge, Smpp, Webadmin and Webbridge CSR

1. Access the CMS CLI with Putty and Log in with the admin account.
2. Run the next commands in order to create CSR for every service needed on CMS. It is also acceptable to create a single cert that has a wild card (*.com) or has the cluster FQDN as CN, FQDNs of each CMS server, and join URL if necessary.

Service	**Comand**
Webadmin	pki csr <cert name> CN:<server FQDN>
Webbridge	pki csr <cert name> CN:<Server FQDN> subjectAltName:<Join Url>,<XMPP domain>
Callbridge **TURN** **Load Balancer**	pki csr <cert name> CN:<Server FQDN's>

3. In case the CMS is clustered, run the next commands.

Service	Command
Callbridge **TURN** **Load Balancer**	pki csr <cert name> CN:<cluster FQDN> subjectAltName:<Peer FQDN’s>
XMPP	pki csr <cert name> CN:<Cluster FQDN> subjectAltName:<XMPP Domain>,<Peer FQDN’s>

Step 3. Generate the Database Cluster CSR and Use Built-in CA to Sign Them

Since CMS 2.7, you are required to have certificates for your database cluster. In 2.7, we included a built-in CA that can be used to sign the database certificates.

1. On all cores, run **database cluster remove**.
2. On the Primary, run **pki selfsigned dbca CN**. Example:**Pki selfsigned dbca CN:tplab.local**
3. On the Primary, run **pki csr dbserver CN:cmscore1.example.com subjectAltName**. Example:**cmscore2.example.com,cmscore3.example.com**.
4. On the Primary, create a cert for database client**pki csr dbclient CN:postgres**.
5. On the Primary, use dbca to sign the dbserver cert**pki sign dbserver dbca**.
6. On the Primary, use dbca to sign the dbclient cert **pki sign dbclient dbca**.
7. Copy the dbclient.crt to all servers that need to connect to a database node
8. Copy the dbserver.crt file to all of the servers that are been joined to the database (nodes that make up the database cluster).
9. Copy the dbca.crt file to all of the servers.
10. On the Primary DB server, run **database cluster certs dbserver.key dbserver.crt dbclient.key dbclient.crt dbca.crt**. This uses the **dbca.crt** as the **root ca-cert**.
11. On the Primary DB server, run **database cluster localnode a**.
12. On the Primary DB server, run **database cluster initialize**.
13. On the Primary DB server, run **database cluster status.** Must see Nodes: (me): Connected Primary.
14. On all other cores that are Joined to the database cluster, run **database cluster certs dbserver.key dbserver.crt dbclient.key dbclient.crt dbca.crt**.
15. On all cores that are connected (not co-located with a database) to the database cluster, run **database cluster certs dbclient.key cbclient.crt dbca.crt**.
16. On cores that are Joined (co-located with a database):
    - run **database cluster localnode a**.
    - run **database cluster join**.
17. ON cores that are connected (not co-located with a database):
    - run **database cluster localnode a**.
    - run **database cluster connect**.

Step 4. Verify the Signed Certificates

- The certificate validity (expiry date) can be verified with certificate inspection, run the command **pki inspect <filename>**.
- You can validate that a certificate matches a private key, run the command **pki match <keyfile> <certificate file>**.
- In order to validate that a certificate is signed by the CA and that the certificate bundle can be used to

assert it, run the command **pki verify <cert> <certificate bundle/Root CA>**.

Step 5. Apply Signed Certificates to Components on CMS Servers

1. In order to apply certificates to Webadmin, run the next commands:

```
webadmin disable
webadmin certs <keyfile> <certificate file> <certificate bundle/Root CA>
webadmin enable
```

2. In order to apply certificates to Callbridge, run the next commands:

```
callbridge certs <keyfile> <certificate file> <certificate bundle/Root CA>
callbridge restart
```

3. In order to apply certificates to Webbridge, run the next commands:

```
webbridge disable
webbridge certs <keyfile> <certificate file> <certificate bundle/Root CA>
webbridge enable
```

4. In order to apply certificates to XMPP, run the next commands:

```
xmpp disable
xmpp certs <keyfile> <certificate file> <certificate bundle/Root CA>
xmpp enable
```

5. In order to apply certificates to the database or replace expired certificates on the current DB cluster, run the next commands:

```
database cluster remove (on all servers, noting who was primary before beginning)
database cluster certs <server_key> <server_certificate> <client_key> <client_certificate> <Root ca_crt
database cluster initialize     (only on primary node)
database cluster join <FQDN or IP of primary>    (only on slave node)
database cluster connect <FQDN or IP of primary> (only on nodes that are not part of the database clust
```

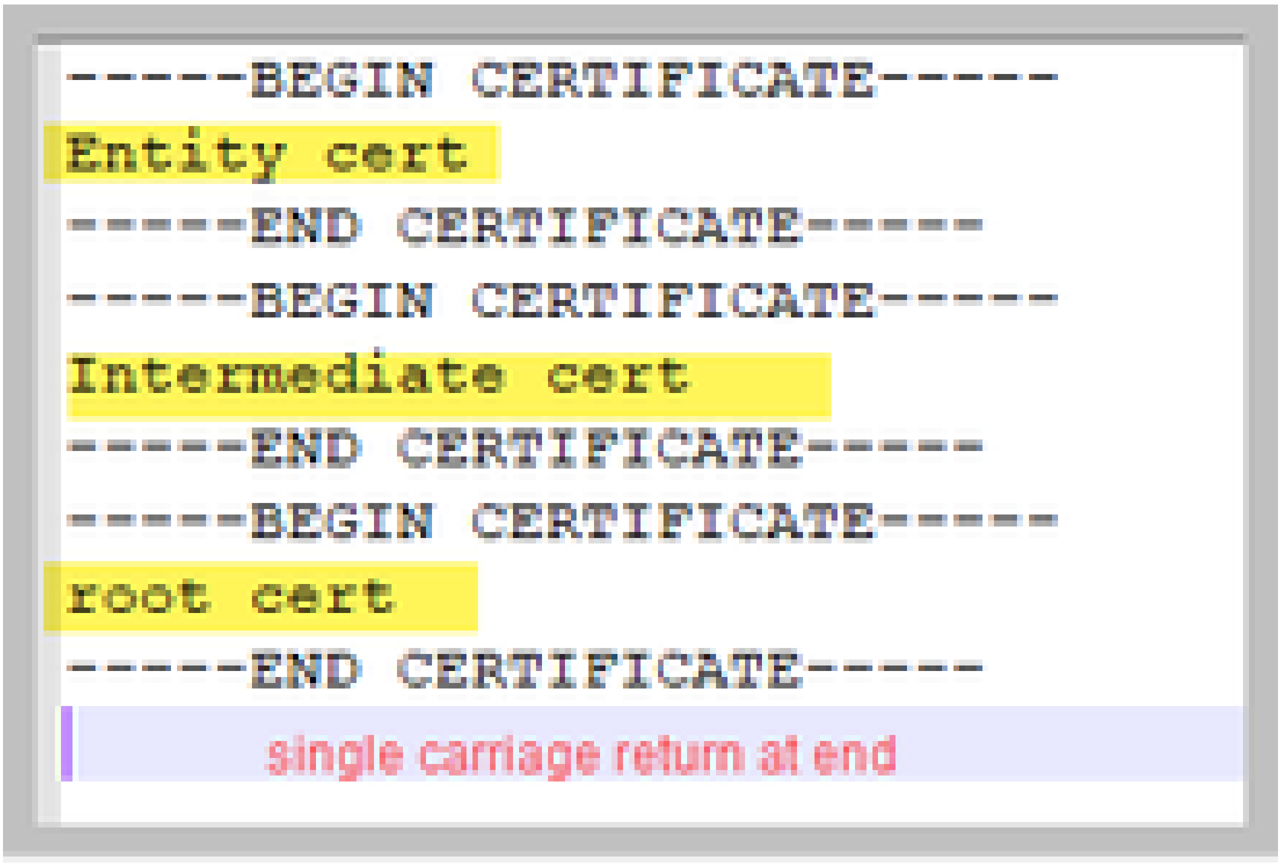
6. In order to apply certificates to TURN, run the next commands:

```
turn disable
turn certs <keyfile> <certificate file> <certificate bundle/Root CA>
turn enable
```

Certificate Trust Chains and Bundles

Since CMS 3.0, you are required to use certificate trust chains or full chain trusts. Also, it is important for any service that you recognize how certs are to be built when you make bundles.

When you build a certificate trust chain, as required for Web bridge 3, you must build it as shown in the image, with the entity cert on top, the intermediates in the middle, the root CA at the bottom, then a single carriage return.



```
-----BEGIN CERTIFICATE-----
Entity cert
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate cert
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
root cert
-----END CERTIFICATE-----
```

single carriage return at end

Anytime that you create a bundle, the certificate must have only one carriage return at the end.

CA bundles would be the same as shown in the image, only, of course, there would be no entity certificate.

Troubleshoot

If you need to replace an expired certificate for all services, except database certificates, the easiest method is to upload new certs with the SAME name as the old certificates. If you do this, the service just needs to be restarted, and you do not need to reconfigure the service.

If you perform **pki csr ...** and that cert name matches a current key, it immediately breaks the service. If the production is live, and you proactively create a new CSR and Key, use a new name. You can rename the currently active name before upload the new cert to the servers.

If the database certificates have expired, you need to check with **database cluster status** who the database Primary is, and on all nodes, run the command **database cluster remove**. Then you can use the instructions from Step 3. Generate the database cluster CSR and use a built-in CA to sign them**.**

Note: In case you need to renew the Cisco Meeting Manager (CMM) certificates, please refer to the next video: Updating the Cisco Meeting Management SSL Certificate.

Related Information

- **Cisco Technical Support & Downloads**