

# Understand Certificate Information to Create a Chain for 9800 WLC

## Contents

---

### [Introduction](#)

### [Prerequisites](#)

[Requirements](#)

[Components Used](#)

### [CSR Generation](#)

### [Third Party Certificate](#)

[Decoded Root CA](#)

[Decoded Intermediate CA](#)

[Decoded Device Certificate](#)

---

## Introduction

This document describes how to decode a certificate with well-known online tools & their interpretation to create a certificate chain in the 9800 WLC.

## Prerequisites

### Requirements

Cisco recommends that you have basic knowledge of these topics:

- Cisco Catalyst 9800 Wireless LAN Controller (WLC)
- Digital Certificate, Certificate Signing Request (CSR) concept.
- OpenSSL software.

### Components Used

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

- OpenSSL software in 1.1.1w version
- Windows computer

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.

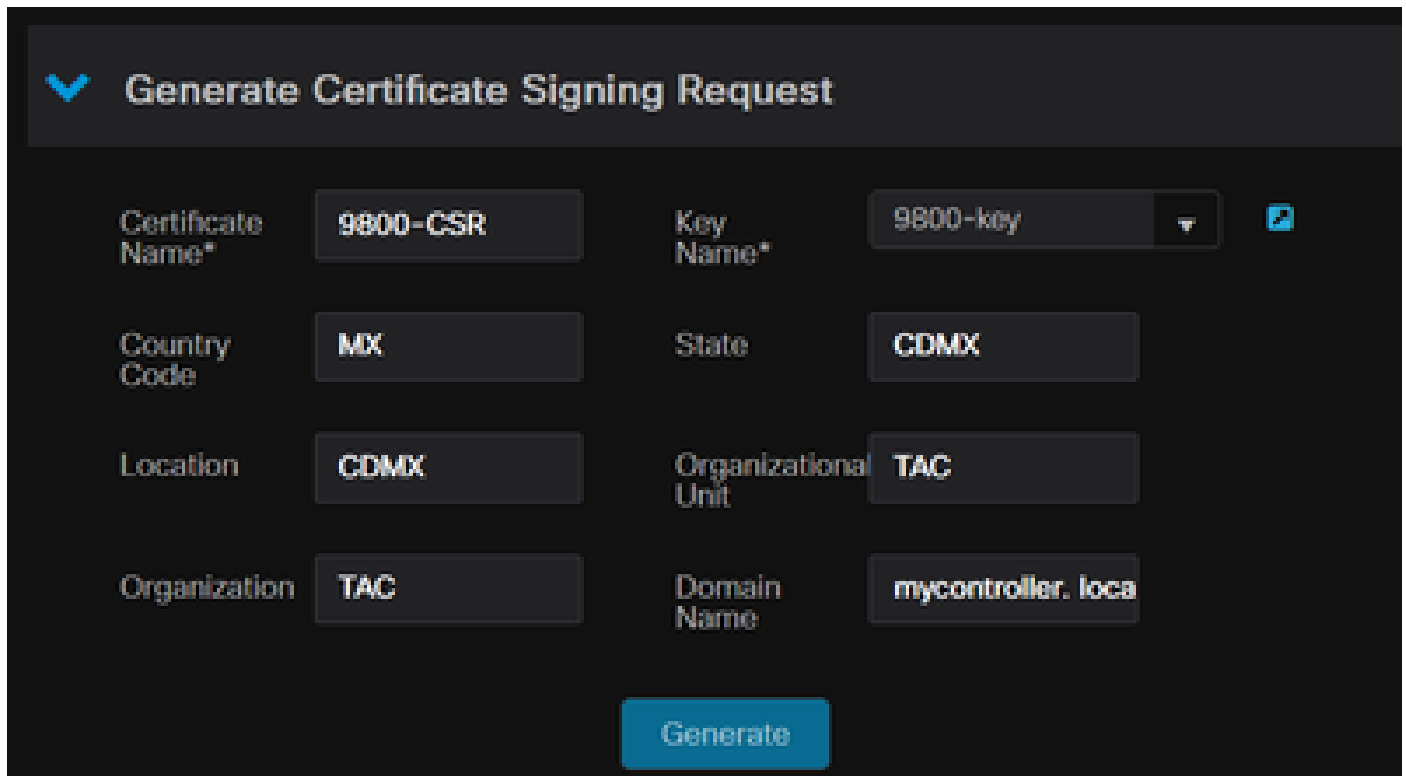
## CSR Generation

The CSR can be generated in the controller or with OpenSSL.

To generate a CSR in the 9800 WLC navigate to **Configuration > Security > PKI Management > Add**

## Certificate > Generate Certificate Signing Request.

When a **Certificate Signing Request** is generated, information such as a Private Key, Common Name (CN), Country Code, State, Location, Organization and Organization Unit is required.



The screenshot shows a web interface for generating a Certificate Signing Request (CSR). The title is "Generate Certificate Signing Request" with a blue checkmark icon. The form is set against a dark background with light-colored text and input fields. The fields are arranged in two columns:

Certificate Name*	9800-CSR	Key Name*	9800-key
Country Code	MX	State	CDMX
Location	CDMX	Organizational Unit	TAC
Organization	TAC	Domain Name	mycontroller. loca

At the bottom center of the form is a blue button labeled "Generate".

*CSR Generation in WLC*

All the CSR information filled in the request is displayed in the decode.

OpenSSL software is the single source of truth when a certificate is decoded. It shows all the information about it.

To decode a certificate in a Windows or MacBook computer with OpenSSL installed, open the Command Prompt as Administrator and run the command **openssl x509 -in <certificate.crt> -text -noout**. The output is shown as console information.



**Note:** Not all openSSL version are supported in 9800 WLC. Suggested versions are 0.9.8 and 1.1.1w

---

There are other online tools to decode certificates that show the output in a more user-friendly way such as CertLogik and SSL Shopper that are not presented in this document.

Be aware that they use the same OpenSSL command already mentioned to decode the certificates.

## **Third Party Certificate**

The CSR is sent to the Certificate Authority (CA) to have it signed and returned. Download all the certificate chain so that you can upload it to the WLC.

To understand the chain of a certificate, all the files received by CA can be decoded. Ensure they are in Base64 format.

You can receive multiple files from the CA. It depends on the number of Intermediate CA files.

To identify each file, you need to decode it.

When a signed certificate is decoded, the **Issuer** section is added. This refers to the CA that signed the certificate.

If you decode a CSR file which is not signed, the **Issuer** section does not exist because it is not signed yet.

This is an example of a multi-level authorization or chained certificate scenario:

- Root CA
- Intermediate CA Certificate
- Device certificate

## Decoded Root CA

For a Root CA, since is the highest authority of the chain, the **Issuer** and **Subject** must be the same.

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      4c:25:79:7e:57:f3:84:85:42:52:1f:c3:4b:f2:64:3f
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: DC = com, DC = Root, CN = RootCA
    Validity
      Not Before: Apr 11 00:21:30 2024 GMT
      Not After : Apr 11 00:31:30 2029 GMT
    Subject: DC = com, DC = Root, CN = RootCA
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:a2:f5:8e:23:db:7b:09:e2:bf:c5:e0:31:a1:35:
        7b:2f:f8:ed:fc:2f:4d:36:c6:b1:92:4e:80:52:6a:
        1a:82:83:3f:77:06:34:ca:0f:2b:fc:ef:84:85:67:
        40:de:a5:59:99:3d:d1:db:f8:ee:55:72:97:2a:bd:
        7e:c5:05:c6:ec:6a:6d:00:ec:22:d5:ff:6a:cd:31:
        49:a2:f0:8d:85:be:ba:e3:a0:db:31:07:e8:9c:3d:
        d4:a9:ab:bc:73:90:b8:a2:ab:a2:87:0c:1d:ac:42:
        f7:e4:26:49:28:18:93:a0:fd:1f:1a:7d:da:1b:e1:
        60:87:dc:38:ce:b7:95:90:64:3d:2f:2b:bc:6e:d7:
        2c:09:5a:54:11:dd:0e:58:63:b4:50:38:87:ea:28:
        28:32:39:8c:e5:2b:b9:13:38:1f:3a:34:b9:32:33:
        af:86:23:3a:40:38:fe:38:18:0c:67:a7:27:66:ab:
        e3:11:66:25:f1:85:48:54:a8:05:0e:9f:02:64:09:
        4f:63:be:a4:53:d5:d7:41:f0:cd:ad:b7:4c:8b:fd:
        ab:a4:c7:fa:95:05:f9:ef:ed:54:ce:90:28:07:1d:
        94:54:4f:bd:6c:7d:4e:a9:70:84:0b:dc:b3:73:3f:
        af:d9:82:86:94:cf:29:35:53:8b:67:95:d3:00:5c:
        ab:e1
```

*Decoded Root CA*

## Decoded Intermediate CA

For Intermediate CA, since it is signed by the Root CA, the **Issuer** must match with the Root CA CN.

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      70:00:00:00:04:18:9f:53:1e:b0:cc:90:b7:00:00:00:00:00:04
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: DC = com, DC = Root, CN = RootCA
    Validity
      Not Before: Apr 11 00:44:27 2024 GMT
      Not After : Apr 11 00:54:27 2026 GMT
    Subject: DC = com, DC = Root, CN = IntermediateCA
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:f1:c9:2b:1a:53:29:55:6d:bc:82:95:36:38:3a:
        08:a4:9e:dd:81:c4:fc:0a:92:6c:2b:30:82:cd:62:
        4c:91:38:ec:09:06:cc:fb:2b:f6:0f:09:43:d3:5a:
        95:6a:3b:2b:4c:bc:d2:03:05:8e:0b:fd:0a:44:c2:
        b8:c1:55:c0:4c:b5:d8:2d:cb:ab:4d:df:d5:d7:96:
        87:21:ea:45:5b:32:db:bd:78:31:fa:5c:cb:1e:66:
        62:8c:42:ff:3e:15:05:25:4e:bf:cd:5a:d7:3e:fb:
        4a:2f:41:95:e0:37:f1:23:22:47:ee:7e:2e:9e:6f:
        a0:24:fe:07:7d:7c:9b:cb:91:9d:05:b6:73:e4:c1:
        c7:04:86:72:a4:6e:73:db:ca:1a:ee:9b:c1:0c:9a:
        39:46:74:96:f8:6f:80:1e:5f:1a:cc:98:7c:91:be:
        7c:98:8b:0d:08:4c:34:ab:30:9c:a0:02:0a:c4:65:
        75:68:0b:f8:29:ea:92:6b:be:c6:83:19:79:fc:bd:
        91:b9:f0:aa:1c:ed:fe:62:2c:27:d7:3e:8b:e3:db:
        74:31:fe:a3:be:5d:8e:12:03:70:9f:f1:3c:0a:61:
        e0:74:0b:08:00:1b:97:7d:01:dd:c7:24:04:7f:f6:
        7e:18:e3:be:ef:a9:33:5d:47:0f:eb:52:6d:07:10:
        f5:d5
```

*Decoded Intermediate CA*

## Decoded Device Certificate

For the Device Certificate, since it is signed by the Intermediate CA, the **Issuer** must match with the Intermediate CA CN

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      76:00:00:00:03:65:c9:0f:4c:b8:29:d8:71:00:00:00:00:00:03
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: DC = com, DC = Root, CN = IntermediateCA
    Validity
      Not Before: Apr 11 00:56:39 2024 GMT
      Not After : Apr 11 00:56:39 2025 GMT
    Subject: DC = com, DC = Root, CN = Users, CN = Administrator
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:d6:24:8c:93:b4:44:13:48:35:94:98:1e:90:f8:
        1b:fc:18:63:df:0f:2a:05:95:38:22:7c:fc:75:69:
        8a:42:07:a8:f9:8b:5f:9f:f2:08:56:ed:d2:1a:b3:
        51:b8:d7:6b:6b:b1:13:aa:8a:ce:3f:c2:6d:cf:f1:
        98:9b:f5:45:1a:77:28:2f:63:d2:91:0c:8d:79:34:
        c2:02:f5:01:16:31:10:49:5c:51:5c:6d:2f:50:82:
        4c:b9:5a:b6:17:be:b6:1a:59:42:8c:97:3c:32:ef:
        cb:52:c7:28:f6:d0:d2:83:4b:ab:2c:5c:14:e1:6b:
        3e:a9:2c:c3:84:25:3b:24:23:d5:1a:7f:2f:42:08:
        45:ba:5b:c4:47:8d:04:52:12:1b:54:9f:9f:85:25:
        9c:ce:71:79:22:3a:19:99:1a:e4:25:9d:7f:91:f0:
        f2:4e:07:be:39:1f:9f:ed:6d:c1:28:33:66:25:54:
        91:62:0e:d3:03:19:69:cc:61:ac:a4:be:b3:ed:25:
        82:b9:77:85:71:30:f8:f7:53:a3:bd:22:a8:8f:0c:
        a7:97:d9:98:79:48:43:ed:5f:c5:c7:17:d0:cd:06:
        e8:da:d3:9b:0e:9e:04:a9:04:da:03:b3:86:96:0d:
        23:2c:3e:6d:81:04:99:38:15:c2:e9:76:da:79:41:
        db:51
  
```

*Decoded Device Certificate*

In a scenario where more than 1 intermediate CA is used, use the same decode process.

Once the chain order is identified, it can be uploaded to the controller.

The 9800 WLC needs the whole chain in the correct order so the certificate can operate properly.

For subsequent steps to upload a certificate to the controller, refer to [Generate and Download CSR Certificates on Catalyst 9800 WLCs](#).

Ensure that you understand the decode process before continuing. If so, the next steps need to be completed to have a Web Auth, Web Admin, or Management certificate uploaded in a 9800 WLC.