

Oracle® Communications

Cloud Native Core, Certificate Management

User Guide



Release 24.1.0

F90677-01

April 2024

ORACLE®

Copyright © 2023, 2024, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1	Introduction	
1.1	Overview	1
1.2	Reference	2
2	OCCM Architecture	
3	OCCM Supported Features	
3.1	Integration with Certificate Authority	1
3.1.1	Establishing Initial Trust Between OCCM and CA	1
3.1.1.1	MAC Based Trust Establishment	1
3.1.1.2	Certificate Based Trust Establishment	2
3.2	Support for HTTPs Encryption	3
3.3	Accessing OCCM from CNC Console	4
3.4	Managing Issuers	5
3.4.1	Pre-configuration for OCCM Bootstrapping	6
3.4.2	Creating Issuer	7
3.4.3	Updating Issuer	14
3.4.4	Deleting Issuers	15
3.5	Managing Certificates	15
3.5.1	Creating OCCM Certificates	16
3.5.1.1	OCCM Certificate Configuration Modes	21
3.5.2	Create NF Certificates	24
3.5.3	Renew NF Certificates	28
3.5.4	Polling for Certificates	29
3.5.5	Deleting the Certificate Configuration	30
3.6	OCCM Retry on Failure	31
3.7	Network Policies	31
4	Introducing OCCM in an Existing NF Deployment	

5 Accessing OCCM Resources Through Curl and Postman

5.1	Generate Access Tokens	1
5.2	Refresh Access Tokens	2
5.3	Issuer Configuration API Access	2
5.4	Certificate Configuration API Access	5
5.5	Logging API Access	7

6 OCCM Metrics

6.1	occm_config_http_requests_total	1
6.2	occm_config_http_response_total	2
6.3	occm_cmp_requests_total	3
6.4	occm_cmp_responses_total	3
6.5	occm_cert_expiry	4
6.6	occm_cert_status	5
6.7	occm_cmp_cli_durations	5
6.8	occm_cert_request_status_total	6

7 OCCM Alerts

7.1	OccmCertExpiryWithinMinorThreshold	1
7.2	OccmCertExpiryWithinMajorThreshold	2
7.3	OccmCertExpiryWithinCriticalThreshold	2
7.4	OccmCertExpired	3
7.5	OccmServiceDown	4
7.6	OccmMemoryUsageMinorThreshold	4
7.7	OccmMemoryUsageMajorThreshold	5
7.8	OccmMemoryUsageCriticalThreshold	6
7.9	OccmCPUUsageMinorThreshold	7
7.10	OccmCMPFailureMinor	7
7.11	OccmCMPFailureMajor	8
7.12	OccmCMPFailureCritical	9
7.13	OccmFailureMinor	9
7.14	OccmFailureMajor	10
7.15	OccmFailureCritical	11
7.16	OccmRenewBeforeValidityCritical	11
7.17	OCCM Alert and MIB Configuration in Prometheus	12

8 OCCM KPIs

8.1	Certificate Expiry Time	1
8.2	Certificate Readiness Status	2

8.3	CMP Request	3
8.4	CMP Responses	3
8.5	Configuration Requests	3
8.6	Configuration Responses	4
8.7	CPU Usage	4
8.8	Memory Usage	4
8.9	OpenSSL CLI Duration (occm_cmp_cli_durations)	4
8.10	Number of requests sent to the CA	5
8.11	Number of responses received from CA	5
8.12	Number of responses based on response code from CA	5
8.13	Type of request sent to CA	5
8.14	Number of certificates issued by CA	5
8.15	Number of CSRs denied by CA or TLS handshake failures or HTTPs connection failures during CA connection	6
8.16	Error while writing the key, certificate, or chain in the Kubernetes secrets	6
8.17	Unable to access or read from Kubernetes secrets	6
8.18	Check Renewed Certificate	7
8.19	Certificate Error and Warnings	7

My Oracle Support

My Oracle Support (<https://support.oracle.com>) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at <http://www.oracle.com/us/support/contact/index.html>. When calling, make the selections in the sequence shown below on the Support telephone menu:

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
 - For Technical issues such as creating a new Service Request (SR), select **1**.
 - For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Acronyms

The following table lists the acronyms and the terminologies used in the document:

Table Acronyms and Terminologies

Acronym	Definition
3GPP	3rd Generation Partnership Project
API	Application Programming Interface
CCA	Client Credentials Assertions
CMP	Certificate Management Protocol
CNC	Cloud Native Core
CNC Console	Cloud Native Configuration Console
OCCM	Oracle Communications Certificate Management
CA	Certification Authority is a trusted entity that issues Secure Sockets Layer (SSL) certificates. CAs are also called issuer in this document.
DNS	Domain Name Server
EE	End Entity
ECC	Elliptic Curve Cryptography
HSM	Hardware Security Module
IDP	Identity Provider
LCM	Life Cycle Management
PKI	Public Key Infrastructure
RA	Registration Authority
RSA	Rivest-Shamir-Adleman
SAN	Subject Alternative Name
URI	Uniform Resource Indicator
URN	Uniform Resource Name
CMP Identity Key	Private Key used by Certificate Management to sign the CMPv2 requests and establish trust between Certificate Management and CA.
CMP Identity Certificate	Certificate that corresponds to and certifies the CMP Identity Key. It is included in the CMPv2 requests for authentication by CA.

What's New in This Guide

This section lists the documentation updates for Release 24.1.x.

Release 24.1.0 - F90677-01, April 2024

- Added the procedure to log in to CNC Console GUI and access the OCCM instance in the [Accessing OCCM from CNC Console](#) section.
- Updated the sample HTTP request and added a sample HTTPS request in [Issuer Configuration API Access](#).
- Updated the Create Issuer procedure in [Creating NF Certificate Using OCCM - Sample Configuration](#).
- Updated the Elliptic Curve possible values in [Creating OCCM Certificates](#) and [Create NF Certificates](#).
- Updated the [Pre-configuration for OCCM Bootstrapping](#) section with details on TLS Trust Store Secret.
- Updated the create issuer procedure in [Creating Issuer](#).
- Added details on Server URL and TLS Configuration fields in [Updating Issuer](#).

1

Introduction

Oracle Communications Cloud Native core, Certificate Management (OCCM) is an automated solution for managing the certificates needed for Oracle 5G Network Functions (NFs). OCCM constantly monitors and renews the certificates based on their validity or expiry period.

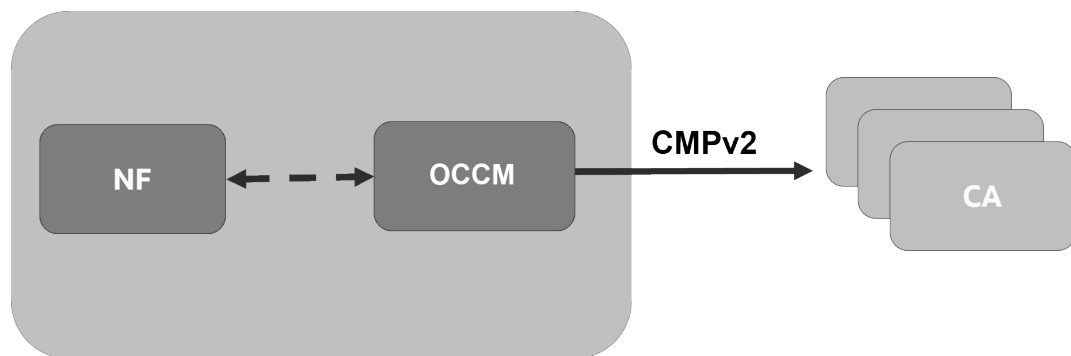
As 3GPP recommends using separate certificates based on the client or server mode and the type of workflow, it leads to many certificates in the network. Automated certificate management eliminates any possibilities of network disruption due to expired certificates. In SBA network deployments, the Network Functions (NFs) are required to support multiple operator certificates for different purposes and interfaces. This amounts to hundreds of certificates in the network with varying validity periods and unwieldy to monitor and renew the certificates manually. Hence, automation of certificate management becomes important to avoid network disruptions due to expired certificates.

1.1 Overview

OCCM integrates with the Certificate Authority(s) using Certificate Management Protocol Version 2 (CMPv2) and RFC4210 to facilitate these certificate management operations:

- Operator-initiated certificate creation
- Operator-initiated certificate recreation
- Automatic certificate monitoring and renewal

Figure 1-1 OCCM Integration with CA



OCCM supports transport of CMPv2 messages using HTTP-based protocol.

OCCM provides the following mechanisms to establish initial trust between OCCM and CA(s):

1. Certificate-based message signing
2. Pre-shared key or MAC based authentication

All the subsequent CMPv2 procedures are authenticated using the certificate-based mechanism in compliance with 3GPP TS 33.310.

The keys and X.509 certificates are managed using Kubernetes secrets.

1.2 Reference

Refer to the following documents for more information:

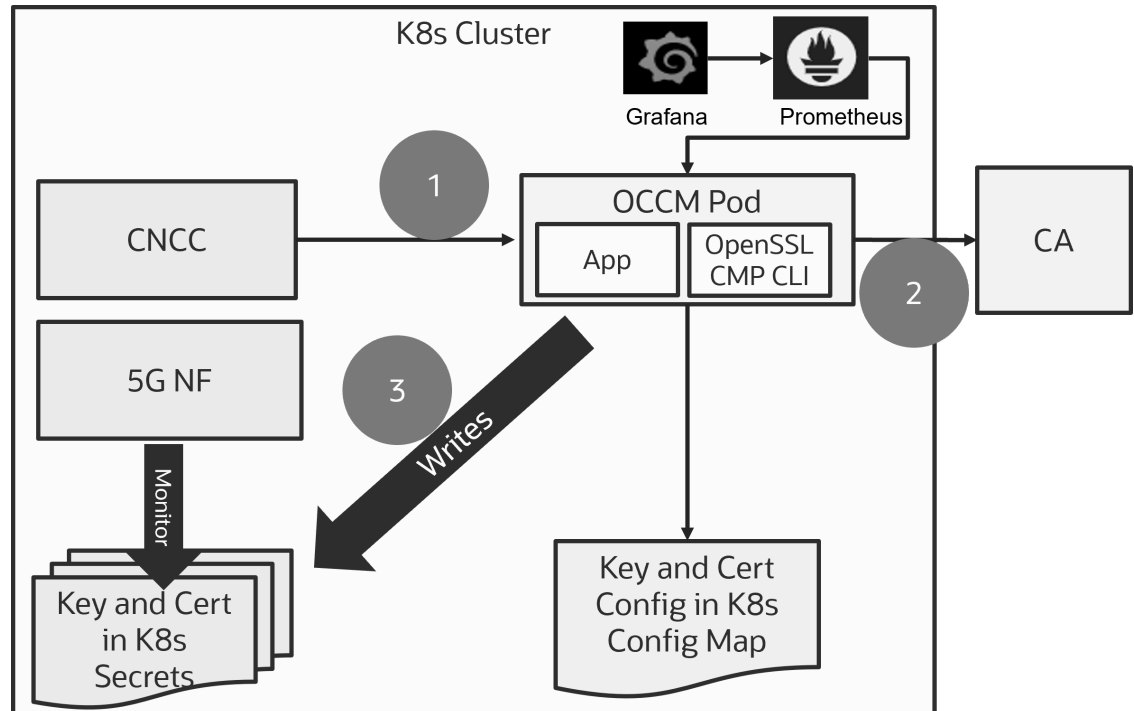
- *Oracle Communications Cloud Native Configuration Console User Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core Security Guide*
- *Oracle Communications Cloud Native Core Solution Upgrade Guide*

2

OCCM Architecture

OCCM is a Cloud Native application consisting of a single microservice. OCCM is packaged and delivered as a CSAR or Helm chart.

Figure 2-1 OCCM Architecture



Architecture Description

OCCM is deployed as a single Kubernetes Pod and has a small resource footprint. The OCCM application uses a set of OpenSSL Certificate Management Protocol (CMP) CLI commands based on the provided configuration and the certificate management procedure that needs to be carried out at a point in time. The Output – Key and Certificate – is stored in configuration defined Kubernetes secret.

Operator provides the desired key and certificate configuration through Console. OCCM contacts the CA for certificate signing. After successful Certificate creation, OCCM writes the key and certificate in Kubernetes secrets.

OCCM provides the following deployment models to support certificate management for the integrated NF(s) instantiated within the same cluster:

- Dedicated deployment model - OCCM resides in the same Kubernetes namespace as the NF or Components.
- Shared deployment model - OCCM is deployed in a separate Kubernetes namespace and can manage certificates of multiple NFs or components deployed in other Kubernetes namespaces.

Appropriate permissions must be assigned to OCCM using Kubernetes Service Account, Role and Role Binding, based on the selected deployment model.

3

OCCM Supported Features

This section describes the features supported by Oracle Communications Cloud Native core, Certificate Management (OCCM).

3.1 Integration with Certificate Authority

OCCM integrates with one or more Certificate Authorities (CAs) using the Certificate Management Protocol version 2 (CMPv2), as proposed by the 3GPP TS33.310. Operators have the flexibility to configure OCCM to integrate with a single CA or multiple CAs, depending on the layout of CA hierarchy deployed in the network. However, it is recommended that each intermediate CA manage multiple certificates of the same type.

The two CMPv2 procedures used by OCCM are:

- Initialization procedure: This is used to create certificates.
- Key update procedure: This is used for certificate renewal scenarios.

OCCM employs two modes to establish initial trust between OCCM and CAs for initial trust establishment:

- Using a pre-shared key
- Using a key and certificate

These options are available when the first request is made towards the CA. For all subsequent requests, OCCM uses the certificate based mechanism to sign the CMPv2 requests in compliance with 3GPP standards.

Note

OCCM supports HTTP 1.0 and HTTP 1.1 versions. OCCM initiates the request using HTTP 1.0. If the CA supports HTTP 1.1 only, then OCCM shifts to using HTTP 1.1 version.

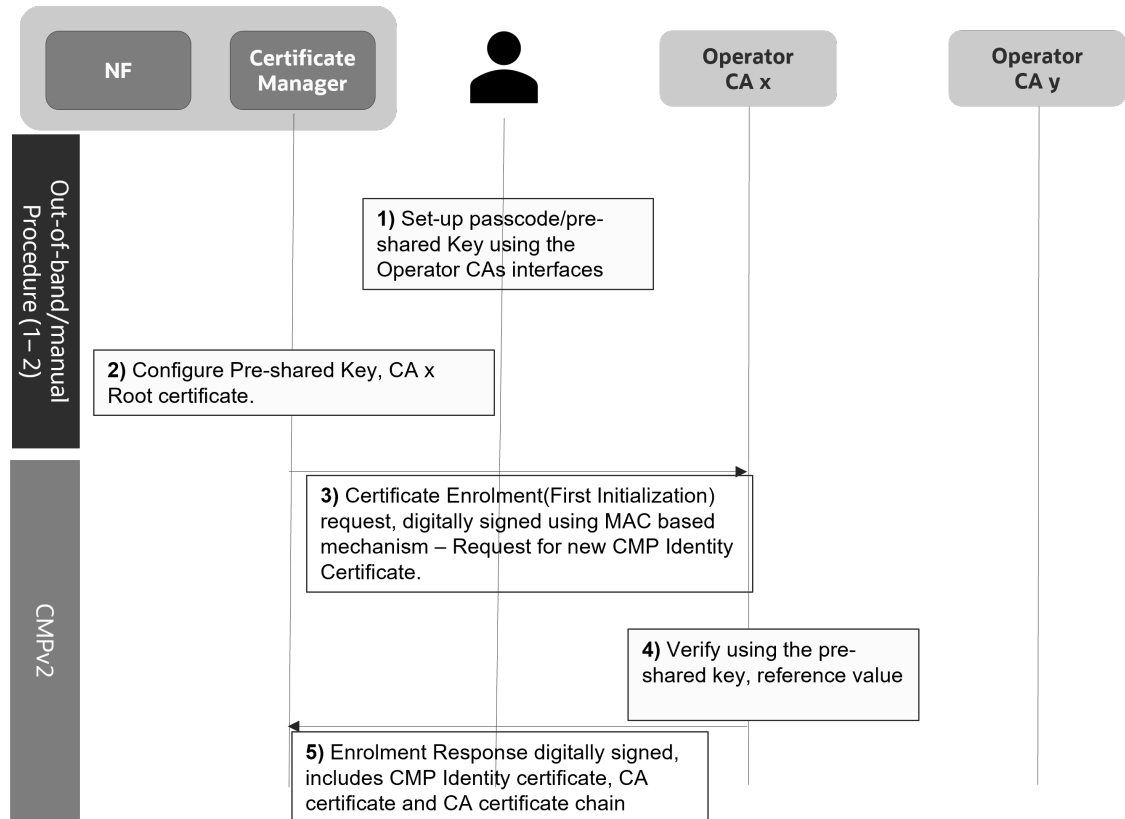
3.1.1 Establishing Initial Trust Between OCCM and CA

OCCM can be configured to establish trust between OCCM and Certificate Authorities (CAs) by enabling PKI message protection in the following ways:

- MAC based trust establishment
- Certificate based trust establishment

3.1.1.1 MAC Based Trust Establishment

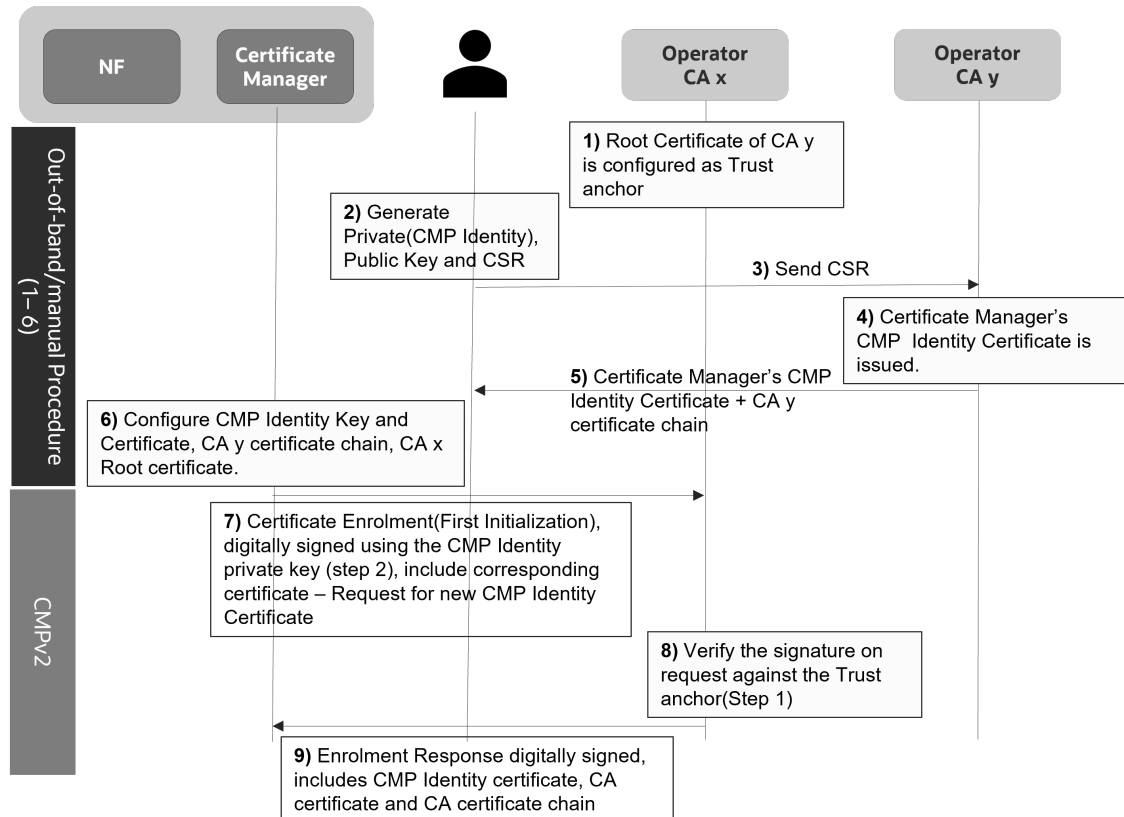
OCCM supports initial trust establishment with each of the configured CAs using the preconfigured pre-shared (MAC) key.

Figure 3-1 MAC Based Trust Establishment

OCCM generates the key pair and requests for the OCCM certificate for each of the configured CAs using the first Initialization Request. The first Initialization Request towards each of the CAs is signed using the preshared key. The CA authenticates the initialization request and signs the OCCM Certificate. OCCM can be configured to authenticate the responses of the first initializing procedure using the preshared (MAC) key. All subsequent requests are always signed using the OCCM key and certificate.

3.1.1.2 Certificate Based Trust Establishment

OCCM supports initial trust establishment with CA using the preconfigured private key and x.509 certificate.

Figure 3-2 Certificate Based Trust Establishment

OCCM signs the first initialization request towards a CA using preconfigured key or certificate.

OCCM can be configured to:

- Continue using the same key and certificate to sign the subsequent CMPv2 requests OR
- Generate a new certificate using the first Initialization Request

In case OCCM uses the same key and certificate to sign the subsequent CMPv2 requests, OCCM requests for generation of the NF certificate in the first Initialization Request.

In case OCCM generates a new certificate using the first Initialization Request, OCCM requests for generation of OCCM certificate in the first Initialization Request. NF Certificate generation is requested from next Initialization Request onwards.

3.2 Support for HTTPs Encryption

Managing HTTPs Encryption

This feature enables you to encrypt the traffic between OCCM and CAs using HTTPs. HTTPs encryption at the transport layer adds an additional layer of security.

OCCM, as a HTTP Client, supports HTTPs connections with CAs using One-Way TLS when authenticating the identity of the CAs. OCCM manages a TrustStore (CA Bundle) to validate the certificates presented by the CAs in the certificate message of the TLS handshake procedure. You can either use the same CA Bundle configuration for all the configured CAs, or different CA Bundles as per your requirements.

OCCM validates the CA certificate as per the RFC 5280 standards, and the TLS handshake can get rejected if the certificate is invalid, or expired:

- Certificate Path validation
- Certificate expiry
- Certificate Strict checking

OCCM supports the following TLS configurations:

- Version TLSv1.2 and TLSv1.3 including support for version rollback to TLSv1.2 in case the CA does not support TLSv1.3
- OCCM acts as the HTTP(s) client while communicating with CA and all the relevant requirements apply.

Configuring HTTPs Encryption

The HTTPs functionality can be manually configured by the operator. The operator can:

- Configure and manually update the CA Bundle used to validate the TLS handshake.
- Enable and disable the strict checking of the X.509 certificates presented for HTTPs. This verifies if the certificates are RFC 5280 compliant.
- Enable or disable the checking of X.509 certificate critical extensions.

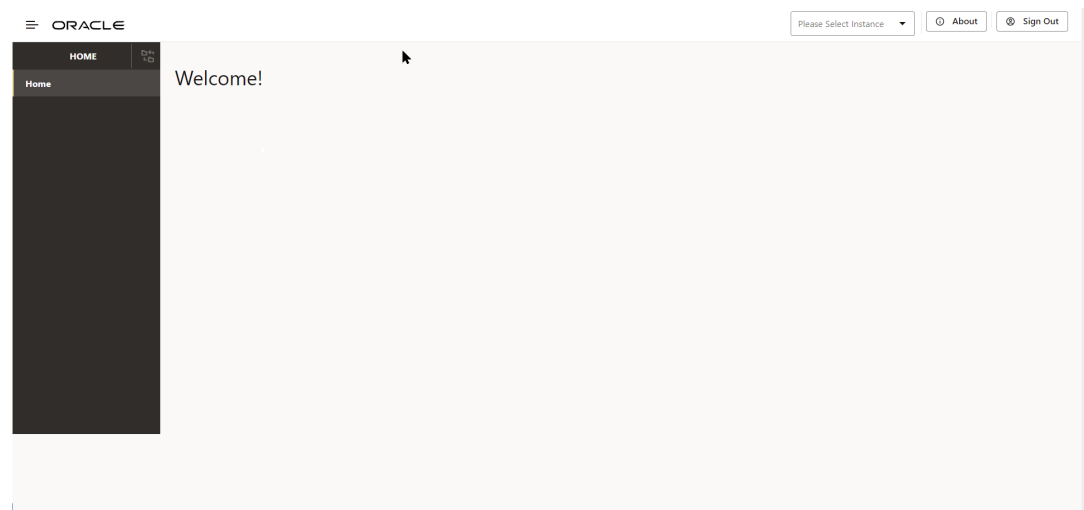
3.3 Accessing OCCM from CNC Console

This section describes the procedure to access the OCCM cluster from the CNC Console GUI.

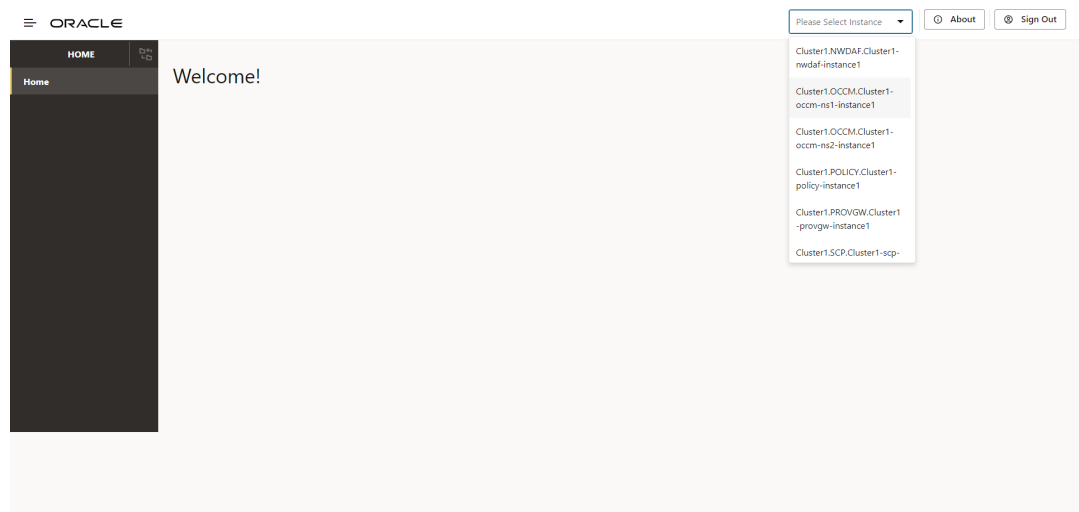
To access OCCM from CNC Console:

1. Log in to CNC Console using your login credentials.

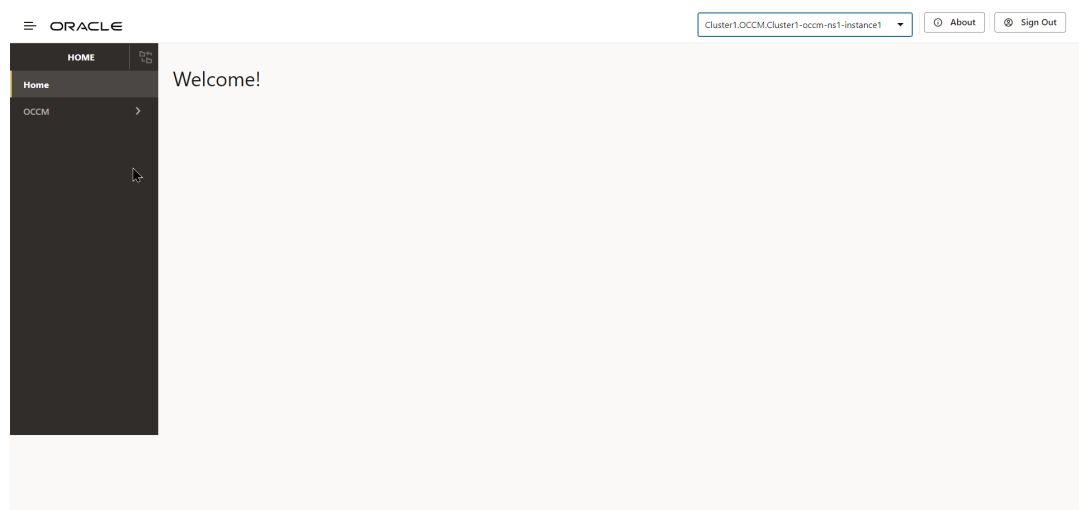
Figure 3-3 CNC Console Landing Page



2. From the **Select Instance** drop-down, select the **OCCM** Instance.

Figure 3-4 OCCM Instance

The OCCM menu appears on the left pane.

Figure 3-5 OCCM Configuration Options

3.4 Managing Issuers

Issuers, also called Certificate Authorities (CAs), are a trusted entity that issues Secure Sockets Layer (SSL) certificates. OCCM supports the following aspects of issuer management:

- Pre-configuration for OCCM Bootstrapping
- Creating Issuers
- Updating Issuers
- Deleting Issuers

3.4.1 Pre-configuration for OCCM Bootstrapping

The following secrets can be pre-configured for OCCM bootstrapping:

- **MAC Secret:** The MAC secret is a manually configured pre-shared key or password based MAC secret and reference. This is used by OCCM to sign the first initialization request. CA then validates the request and issues a signed OCCM certificate. For more information see the 'Using the pre-shared key' section in [OCCM Certificate Configuration Modes](#). To create the MAC Secret, run the following command:

```
kubectl create secret generic <k8s secret name> --from-literal=<mac secret key>=<mac secret value> --from-literal=<reference key>=<reference value> -n <namespace>
```

For example:

```
kubectl create secret generic ca1-mac-secret --from-literal=pwd='pass:****' --from-literal=ref='abcd' -n ns1
```

- **CMP Identity Secret:** The CMP Identity secret is a manually configured private key and certificate, using which OCCM certificate is requested from CA. This is used by OCCM to sign the first initialization request. CA then validates the request and issues a signed OCCM certificate. You can also use the same private key and certificate as OCCM certificate. For more information, see the 'Using the pre-configured private key and certificate' section in [OCCM Certificate Configuration Modes](#). To create the CMP Identity Key, run the following command:

```
kubectl create secret generic <k8s secret name> --from-file=<cmp key file location> --from-file=<cmp cert file location> -n <namespace>
```

For example:

```
kubectl create secret generic ca1-cmp-identity-secret --from-file=cmpkey.pem --from-file=cmpcert.pem -n ns1
```

- **OCCM Trust Store Secret:** The OCCM Trust Store secret holds OCCM trust store information (CA certificates), and is used as a trust anchor when validating the digital signature included in the CMP responses.

To create the OCCM Trust Store secret, run the following command:

```
kubectl create secret generic <k8s secret name> --from-file=<CA root cert file location> --from-file=<Intermediate CA cert file location> --from-file=<CMP server cert file location> -n <namespace>
```

For example:

```
kubectl create secret generic ca1-occm-trust-store-secret --from-file=caroot.pem --from-file=intcacert.pem --from-file=servercert.pem -n ns1
```

- **TLS Trust Store Secret:** If TLS is enabled for issuer, TLS Trust Store secret should be provided else it should be skipped. It holds the CA certificates to be used as trust anchors

when authenticating the TLS server certificate. To create the TLS Trust Store secret, run the following command:

```
kubectl create secret generic <k8s secret name> --from-file=<CA cert file location> -n <namespace>
```

For example:

```
kubectl create secret generic ca1-tls-trust-store-secret --from-file=caroot1.pem -n ns1
```

HTTPS communication between OCCM and CA

OCCM supports HTTPS connections with CA using one-way TLS. To enable the same, the operator has to set `enableTLS` option in the issuer configuration to `true` and configure the HTTPS scheme server URL. TLS trust store has to be configured with trust anchors in order to authenticate the TLS server.

In order to enable or disable strict checking of the X.509 certificates presented for HTTPS, the following deployment time (helm) parameters can be configured.

- **occmConfig.cmp.config.tls.enableX509StrictCheck:** This field when set to `true` enables strict checking of the X.509 certificates presented for HTTPS. Errors are thrown for the certificates which are not compliant with RFC 5280.
- **2. occmConfig.cmp.config.tls.ignoreCriticalExtensionsCheck:** This field when set to `true` ignores checking of the critical extensions in X.509 certificates presented for HTTPS.

Normally if an unhandled critical extension is present that is not supported by OpenSSL the certificate is rejected in compliance with RFC 5280.

Note

This configuration will be applied only when TLS is enabled for an issuer.

3.4.2 Creating Issuer

Issuers are resources that represent CAs and are able to generate signed certificates. You can configure issuers through REST API or using the CNC Console GUI. The maximum number of issuers that can be supported at a time is 30.

Configuring Issuer Using CNC Console GUI

To manually configure issuer using CNC Console GUI:

1. Log in to CNC Console using your login credentials and select the OCCM Instance.
2. Click **OCCM** from the left pane and then click **Issuer**.
3. Click **Add**. The **Create Issuer** page appears.

Figure 3-6 Create Issuer

Create Issuer

UUID:

UUID

Name:

Name

Server URL:

Server URL

Recipient Distinguished Name:

Recipient Distinguished Name

Issuer Distinguished Name:

Issuer Distinguished Name

Total Timeout (Seconds):

Total Timeout (Seconds)
720

Message Timeout (Seconds):

Message Timeout (Seconds)
120

4. Enter the following information on the Create Issuer page:

Table 3-1 Create Issuer

Field Name	Description
Name	Name of the Issuer.
Recipient Distinguished Name	Distinguished name(DN) of the CMP server (usually the addressed CA) used in the recipient field of CMP request message headers. The argument must be formatted as / type0=value0/type1=value1/type2=.... Special characters may be escaped by \ (backslash); whitespace is retained. Empty values are permitted, but the corresponding type will not be included. Giving a single / will lead to an empty sequence of RDNs (a NULL-DN). Multi-valued RDNs can be formed by placing a + character instead of a / between the AttributeValueAssertions (AVAs) that specify the members of the set. For example: /DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe
Server URL	Domain URL of CA
Issuer Distinguished Name	X509 issuer Distinguished Name of the CA server to place in the requested certificate template in IR or KUR. The argument must be formatted as / type0=value0/type1=value1/type2=.... Special characters may be escaped by \ (backslash); whitespace is retained. Empty values are permitted, but the corresponding type will not be included. Giving a single / will lead to an empty sequence of RDNs (a NULL-DN). Multi-valued RDNs can be formed by placing a + character instead of a / between the AttributeValueAssertions (AVAs) that specify the members of the set. For example: /DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe
Total Timeout (Seconds)	The total time in seconds allowed for the CMP transaction to complete.

Table 3-1 (Cont.) Create Issuer

Field Name	Description
Message Timeout (Seconds)	The total time (in seconds) a CMP request-response message round trip is allowed to take.

5. Under **CMP Client Authentication Options For OCCM certificate**, enter the following information:

Table 3-2 Initial Authentication Options

Field Name	Possible Values
Type	MAC, SIGNATURE For more information, see OCCM Certificate Configuration Modes .
Digest Algorithm	SHA256, SHA384, SHA512
MAC Algorithm	HMACSHA256, HMACSHA384, HMACSHA512

Figure 3-7 CMP Client Authentication Options For OCCM Certificate

Figure 3-7 shows a configuration panel titled "CMP Client Authentication Options For OCCM Certificate". It contains three dropdown menus:

- Type:** The dropdown menu is open, showing "Type" as the selected option.
- Digest Algorithm:** The dropdown menu is open, showing "Digest Algorithm" as the selected option.
- MAC Algorithm:** The dropdown menu is open, showing "MAC Algorithm Please Select" as the selected option.

6. If you are using the password based MAC authentication mechanism, then under **MAC Authentication Input**, enter the following information:

Table 3-3 MAC Authentication Input

Field Name	Description
Namespace	Name of the Kubernetes namespace.
Secret Name	Kubernetes secret name.
Password Key	Kubernetes secret data key against which MAC secret is provided.
Reference Key	Kubernetes secret data key against which reference string is provided.

Figure 3-8 MAC Authentication Input

MAC Authentication Input

Namespace:

Namespace

Secret Name:

Secret Name

Password Key:

Password Key

Reference Key:

Reference Key

7. Under **Signature Authentication Input**, enter the following information:

Table 3-4 Signature Authentication Input

Field Name	Description
Namespace	Name of the Kubernetes namespace.
Secret Name	A unique secret name.
Key	Kubernetes secret data key against which the pre-configured private key file (private key file for the client's current CMP signer certificate) is provided.
Cert	Kubernetes secret data key against which the pre-configured certificate (client's current CMP signer certificate) is provided.
Extra Certs	Extra Certificates, if any, for client authentication.

Figure 3-9 Signature Authentication Input

Signature Authentication Input

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

Cert:

Cert

Extra Certs:

Extra Certs

8. Under **CMP Client Authentication Options For Other certificate**, enter the following information:

Table 3-5 CMP Client Authentication Options For Other certificate

Field Name	Possible Values
Type	SIGNATURE
Digest Algorithm	SHA256, SHA384, SHA512

Figure 3-10 CMP Client Authentication Options For Other certificate

▼ CMP Client Authentication Options For Other certificate

Type:

Type▼

Digest Algorithm:

Digest Algorithm▼

9. Under **Signature Authentication Input**, enter the following information:

Table 3-6 Signature Authentication Input

Field Name	Description
Namespace	Name of the Kubernetes namespace
Secret Name	A unique secret name
Key	Kubernetes secret data key against which OCCM key is provided or created based on whether OCCM certificate is created in manual or automatic mode.
Cert	Kubernetes secret data key against which OCCM certificate is provided or created based on whether OCCM cert is created in manual or automatic mode.
Extra Certs	List of Kubernetes secret data keys against which the certificates to append in the extraCerts field can be provided or will be created (if received from CA) along with the OCCM certificate, based on whether OCCM cert is created in manual or automatic mode.

Figure 3-11 Signature Authentication Input

Signature Authentication Input

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

Cert:

Cert

Extra Certs:

Extra Certs

10. Under **Occm Trust-Store Secret Input**, enter the following information:

Table 3-7 Occm Trust-Store Secret Input

Field	Description
Namespace	Name of the Kubernetes namespace.
Name	Kubernetes secret which holds OCCM trust store information (CA certificates).
Root CA Certs	The certificate(s), typically of root CAs, the client uses as trust anchors when validating the certificate issued by CA. Note: If server certificate is present, this is ignored.
Intermediate CA Certs	Any untrusted intermediate CA certificate(s) to use when validating newly enrolled certificates.
Server Cert	CMP server or CA server's certificate to expect and directly trust when validating the certificate issued by CA. Note: If this is present, root CA certificates will be ignored.

Figure 3-12 Occm Trust-Store Secret Input

▼ Occm Trust-Store Secret Input

Namespace:

Namespace

Name:

Name

Root CA Certs:

Root CA Certs

Intermediate CA Certs:

Intermediate CA Certs

Server Cert:

Server Cert

11. Enter either the root CA certificates and intermediate CA certificate, or the server certificate in the respective fields.
12. Under TLS Configuration, enter the following information:

Field	Description
Enable TLS	When set to true, HTTPS connection to CA is made. Ensure selecting scheme as HTTPS in server URL if this is set to true.

Figure 3-13 TLS Configuration

Server URL:

Server URL

https://ca1-openssl.mock.svc.thrust5:8443

Field	Description
Namespace	Kubernetes namespace where TLS trust store secret is present.
Name	Kubernetes secret which holds TLS trust store information (CA certificates).
TLS trusted Certs	Trusted certificate(s) to use for validating the TLS server certificate.

Figure 3-14 Enable TLS

▼ TLS Config

Enable TLS:

▼ TLS Trust-Store Secret Input

Namespace:

Namespace
ns1

Name:

Name
ca1-tls-trust-store-secret

TLS Trusted Certs:

TLS Trusted Certs

ManagementCA.pem X

13. Click **Save**.

3.4.3 Updating Issuer

You can update all the fields in Edit issuer if no certificate configuration is attached to it. However, if any certification configuration is mapped to the given issuer, only the following fields can be edited:

- Server URL
- TLS Configuration

To update the issuer:

1. Log in to CNC Console using your login credentials and select the OCCM Instance.
2. Click **OCCM** from the left pane and then click **Issuer**.

Figure 3-15 Issuer Page

Issuer

Type to Filter

Refresh

Add

UUID	Name	Server URL	Actions
529c446b-1294-423f-89b6-1bea61dbb447	ca21	http://ca-21-openssl-mock.occmcc-thrust5-01.svc.thrust5.8080	
9e601658-cab9-4a4a-881e-d6f5c38791223	CA1	http://ca-0211-openssl-mock.occmcc-thrust5-01.svc.thrust5.8080	
0f19d193-ff06-461d-aa10-a707a6b8acc9	Test-CA	http://test-ca-openssl-mock.occmcc-thrust5-01.svc.thrust5.8080	

3. Click the pencil icon next to the issuer that you want to update. The Edit Issuer page appears.

Figure 3-16 Edit Issuer

Edit Issuer	
UUID:	UUID 529c446b-1294-429f-89bd-1bea51dba447
Name:	Name ca21
Server URL:	Server URL http://ca-21-openssl-mock.occncc-thrust5-01.svc.thrust5.0090
Recipient Distinguished Name:	Recipient Distinguished Name /CN=test
Issuer Distinguished Name:	Issuer Distinguished Name /CN=test
Total Timeout (Seconds):	Total Timeout (Seconds) 720
Message Timeout (Seconds):	Message Timeout (Seconds) 120
▼ CMP Client Authentication Options For OCCM Certificate	
Type:	Type SIGNATURE
Digest Algorithm:	Digest Algorithm SHA256
▼ Signature Authentication Input	
Namespace:	Namespace occncc-thrust5-01

4. Edit the fields that you need to update and then click **Save**.

Note

Issuer can't be edited if it is in use by any certificate.

3.4.4 Deleting Issuers

To delete issuers:

1. Log in to CNC Console using the login credentials, and select the OCCM Instance.
2. Click **OCCM** from the left pane and then click **Issuer**.
3. Click **Delete** and click **OK** on the confirmation prompt to delete the issuer.

Note

An issuer can only be deleted if there are no certificates referring to this issuer entry.

3.5 Managing Certificates

OCCM creates a new key-pair (private and public key) for each of the certificates to be created. This is applicable to both NF and OCCM certificates.

OCCM supports the following key aspects of certificate management:

- Creating OCCM Certificates
- Creating NF Certificates
- Monitoring and Renewing OCCM and NF Certificates

Note

- Grafana dashboards can be used to visualize certificate status such as expiry time, etc.
- The maximum number of certificates supported (OCCM certificates and NF certificates combined) is 100.

3.5.1 Creating OCCM Certificates

Each certificate configuration in OCCM is a certificate request. It specifies input fields that are used to generate a private key pair and certificate signing request to obtain a signed certificate from the referenced issuer.

To create an OCCM certificate:

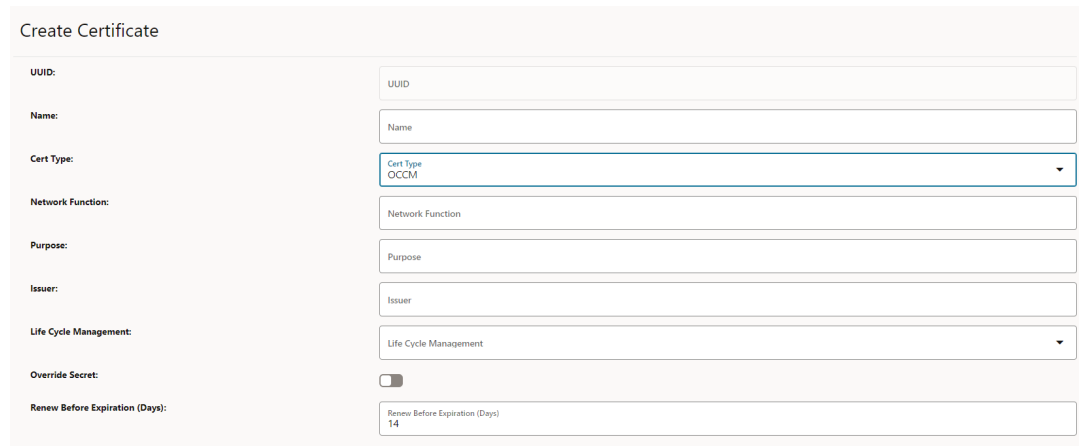
- A CMPv2 Initialization Request (IR) is sent to the CA. Each request supports one certificate request. A separate IR for each certificate request is used.
- The IRs and Certificate Confirms are digitally signed by the CMP Identity Key.
- OCCM supports Proof of Possession (PoP) in the initialization request. PoP of the signing key contains the algorithm identifier and signature. This signature is based on the certificates template structure.
- The recommended signing algorithms for the CMPv2 messages and Proof of Possession are RSA Encryption and ECDSA.
- The recommended hash algorithms for the CMPv2 messages and Proof of Possession are SHA-256 and SHA-384.

When the preshared key mechanism is used to establish the initial trust between OCCM and CA, the first OCCM certificate, also known as CMP Identity Key Certificate, corresponding to a particular CA is created in the first initialization procedure.

When certificate based initial trust is established, then the operator can choose to continue with the preconfigured OCCM certificate, or can choose to create a new OCCM certificate using the first initialization procedure, which is configurable.

To create OCCM Certificates:

1. Login to CNC Console using your login credentials and select the OCCM Instance.
2. Click **OCCM** from the left pane and then click **Certificate**.
3. Click **Add**. The **Create Certificate** page appears.

Figure 3-17 Create OCCM Certificate


Create Certificate

UUID:

Name:

Cert Type:

Network Function:

Purpose:

Issuer:

Life Cycle Management:

Override Secret: ☐

Renew Before Expiration (Days):

4. Enter the following information:

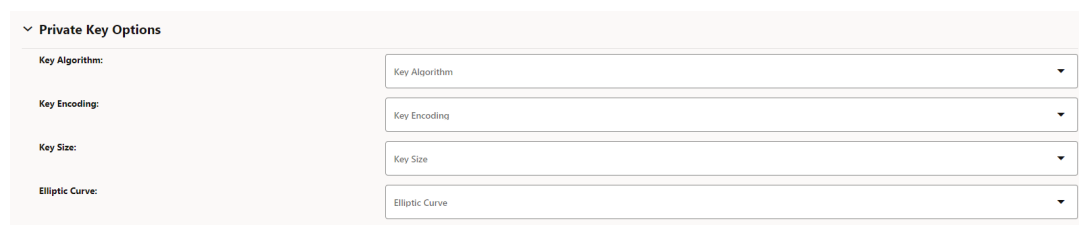
Table 3-8 Create OCCM Certificate

Field Name	Description and Possible Values
Name	Name of the certificate.
Cert Type	Select OCCM for OCCM certificates.
Network Function	OCCM
Purpose	Purpose of the OCCM Certificate.
Issuer	Name of the issuer for the certificate.
Life Cycle Management	Possible values are MANUAL and AUTOMATIC. For more information, see OCCM Certificate Configuration Modes .

5. Under **Private Key Options**, enter the following information:

Table 3-9 Private Key Options

Field Name	Possible Values
Key Algorithm	RSA, EC
Key Encoding	DER, PEM
Key Size	KEYSIZE_2048, KEYSIZE_4096
Elliptic Curve	SECP256r1, SECP384r1

Figure 3-18 Private Key Options


Private Key Options

Key Algorithm:

Key Encoding:

Key Size:

Elliptic Curve:

6. The **Private Key Output** section is auto populated from corresponding issuer after the certificate is saved. You can skip this section.

Table 3-10 Private Key Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Kubernetes Secret Name.
Key	Kubernetes secret key against which the key-pair will be stored.

Figure 3-19 Private Key Output

Private Key Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

7. Under **Public Key Certificate Options**, enter the following:

Table 3-11 Public Key Certificate Options

Field Name	Description
Key Usage	Value(s): DIGITAL_SIGNATURE
Extended Key Usage	Value(s): CLIENT_AUTH and SERVER_AUTH
Basic Constraints	Value(s): END_ENTITY
Subject	Country: Enter country code: State: Enter state code. Location: City or town where company is legally located. Organization: Name of your organization. Organisation Unit: Name of business unit. Common Name: The Common Name (CN) represents the server name to be protected by the certificate. Requested Validity (Days): Number of days requested for which the certificate will be valid.

Figure 3-20 Public Key Certificate Options

Public Key Certificate Options

Key Usage

Critical: ☒

Value(s):

Extended Key Usage

Critical: ☐

Value(s):

Basic Constraints

Critical: ☐

Value:

Subject

Country:

State:

Location:

8. Under **Subject Alternate Names**, enter the following:

Table 3-12 Subject Alternate Names

Field Name	Description
IP Address	The IPs you want to protect under this certificate.
DNS Names	List of DNS domain names.
URI ID API Roots	List of URI IDs.
URI ID URNs	List of URI IDs.

Figure 3-21 Subject Alternate Names

Subject Alternate Names

Critical: ☒

IP Addresses:

DNS Names:

URI ID API Roots:

URI ID URNs:

9. The **Certificate Output** section is auto populated from corresponding issuer after the certificate is saved. You can skip this section.

Table 3-13 Certificate Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Name of the secret.

Table 3-13 (Cont.) Certificate Output

Field Name	Description
Key	The key against which the certificate will be populated.

Figure 3-22 Certificate Output

▼ Certificate Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

10. (Optional) Under **Certificate Chain Output**, enter the following:

Table 3-14 Certificate Chain Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Name of the secret.
Key	Kubernetes secret key against which the certificate chain will be stored.

Figure 3-23 Certificate Chain Output

▼ Certificate Chain Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

If the **Certificate Chain Output** section is filled, then the certificate chain can either be obtained from the CA or can be configured manually. This is based on the `extractCertChainFromCmpResponse` helm parameter. For more information, see *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide*

 Note

`extractCertChainFromCmpResponse`: This field when set true specifies that certificate chain will be extracted from CA's CMP response message. When false, the operator can configure the chain manually. This certificate chain is used in the TLS handshake along with the certificate.

11. Click **Save**.

3.5.1.1 OCCM Certificate Configuration Modes

The following section highlights the configuration applicable to these modes and control how the OCCM certificates are generated. The purpose of the following issuer configuration and certificate configuration sections is to highlight the difference in the fields for different modes.

OCCM can be configured with one of the following modes available to establish the initial Trust with the CA(s):

- Using the pre-shared key
- Using the pre-configured private key and certificate

Using the Pre-shared Key

With this mode of configuration, OCCM signs the first initialization request using the pre-shared key. CA validates the request and issues a signed OCCM certificate.

1. Issuer configuration

To configure the issuer using the pre-shared key,

- The MAC authentication input must be provided under **CMP Client Authentication Options for OCCM Certificate**.

Figure 3-24 CMP Client Authentication Options for OCCM Certificate

CMP Client Authentication Options For OCCM Certificate	
Type	MAC
MAC Algorithm	HMACSHA256
MAC Authentication Input	
Noncespace	Noncespace ns1
Secret Name	Secret Name ca1-mac-secret
Password Key	Password Key pwd
Reference Key	Reference Key ref

- OCCM key and certificate output location must be specified under **CMP Client Authentication Options for Other Certificate**. OCCM certificate received from CA will be written here.

Figure 3-25 CMP Client Authentication Options for Other Certificate

CMP Client Authentication Options For Other certificate	
Type	SIGNATURE
Digest Algorithm	Digest Algorithm SHA256
Signature Authentication Input	
Noncespace	Noncespace ns1
Secret Name	Secret Name ca1-occm-key-cert-secret
Key	Key occmkey.pem
Certs	Cert occmcert.pem
Extra Certs	Extra Certs

2. Certificate configuration

To configure the OCCM Certificate using the pre-shared key, select OCCM from the **Cert Type** drop down, and select AUTOMATIC from the **Life Cycle Management** on the **Create Certificate** page.

Figure 3-26 OCCM Certificate Configuration using Pre-shared Key

UUID:	UUID
Name:	Name
Cert Type:	Cert Type OCCM
Network Functions:	Network Function
Purpose:	Purpose
Issuer:	Issuer
Life Cycle Management:	Life Cycle Management AUTOMATIC

Using the pre-configured private key and certificate

The pre-configured private key and certificate mode can be used in the following two ways:

1. OCCM signs the first initialization request using the pre-configured private key and certificate. CA validates the request and issues a signed OCCM certificate.
 - a. **Issuer Configuration**
Here, to configure the issuer,
 - i. Provide the Signature Authentication Input under **CMP Client Authentication Options for OCCM Certificate**.

Figure 3-27 CMP Client Authentication Options for OCCM Certificate

CMP Client Authentication Options For OCCM Certificate	
Type	Type SIGNATURE
Digest Algorithm	Digest Algorithm SHA256
Signature Authentication Input	
Namespace	Namespace ns1
Secret Name	Secret Name ca1-cmp-identity-secret
Key	Key cmpkey.pem
Certs	Cert cmpcert.pem
Extra Certs	Extra Certs

- ii. OCCM key and certificate output location must be specified under **CMP Client Authentication Options for Other Certificate**. OCCM certificate received from CA will be written here.

Figure 3-28 CMP Client Authentication Options for Other Certificate

CMP Client Authentication Options For Other certificate	
Type	Type SIGNATURE
Digest Algorithm	Digest Algorithm SHA256
Signature Authentication Input	
Namespace	Namespace ns1
Secret Name	Secret Name ca1-occm-key-cert-secret
Key	Key occmkey.pem
Certs	Cert occmcert.pem
Extra Certs	Extra Certs

- b. **OCCM Certificate Configuration**
To configure the OCCM certificate, select OCCM from the **Cert Type** drop down, and select AUTOMATIC from the **Life Cycle Management** on the **Create Certificate** page.
2. The pre-configured private key and certificate (generated out of band) can itself be used as the OCCM certificate.

- a. **Issuer Configuration**
 - i. Here, you must skip the **CMP Client Authentication Options for OCCM Certificate**.

Figure 3-29 Issuer Configuration

CMP Client Authentication Options For OCCM Certificate

Type

Type

Digest Algorithm

Digest Algorithm

MAC Algorithm

MAC Algorithm

MAC Authentication Input

Namespace

Namespace

Secret Name

Secret Name

Password Key

Password Key

Reference Key

Reference Key

Signature Authentication Input

Namespace

Namespace

Secret Name

Secret Name

Key

Key

Certs

Certs

Extra Certs

Extra Certs

- ii. OCCM key and certificate output location must be specified under **CMP Client Authentication Options for Other Certificate**. Manually created OCCM key and certificate location should be specified here.

Figure 3-30 CMP Client Authentication Options for Other Certificate

CMP Client Authentication Options For Other certificate

Type

Other SIGNATURE

Digest Algorithm

Digest Algorithm SHA256

Signature Authentication Input

Namespace

Namespace test

Secret Name

Secret Name call-cmp-identity-secret

Key

Key call-cmpkey.pem

Certs

Certs cmpcert.pem

Extra Certs

Extra Certs

- b. **OCCM Certificate Configuration**
To configure the OCCM certificate, select OCCM from the **Cert Type** drop down, and select **MANUAL** from the **Life Cycle Management** on the **Create Certificate** page.

Figure 3-31 OCCM Certificate Configuration

UUID:

UUID

Name:

Name

Cert Type:

Cert Type OCCM

Network Functions:

Network Function

Purpose:

Purpose

Issuer:

Issuer

Life Cycle Management:

Life Cycle Management MANUAL

Note

This configuration is available for each of the issuers, therefore the modes for the CAs can be controlled individually.

3.5.2 Create NF Certificates

To create an NF certificate:

- A CMPv2 Initialization Request (IR) is sent to the CA. Each request supports one certificate request. A separate initialization request for each certificate request is used.
- The IRs and Certificate Confirms are digitally signed by the CMP Identity Key.
- OCCM supports Proof of Possession (PoP) in the initialization request. PoP of the signing key contains the algorithm identifier and signature. This signature is based on the certificates template structure.
- The recommended signing algorithms for the CMPv2 messages and Proof of Possession are RSA Encryption and ECDSA.
- The recommended hash algorithms for the CMPv2 messages and Proof of Possession are SHA-256 and SHA-384.

You can configure NF certificates through REST API or using the CNC Console GUI.

To create NF Certificates using CNC Console GUI:

1. Log in to CNC Console using your login credentials and select the OCCM Instance.
2. Click **OCCM** from the left pane and then click **Certificate**.
3. Click **Add**. The **Create Certificate** page appears.

Figure 3-32 Create NF Certificate

Create Certificate

UUID:	
Name:	
Cert Type:	
Network Function:	
Purpose:	
Issuer:	
Life Cycle Management:	
Override Secret:	☐
Renew Before Expiration (Days):	14
▼ Private Key Options	
Key Algorithm:	
Key Encoding:	
Key Size:	
Elliptic Curve:	

4. Enter the following information:

Table 3-15 Create NF Certificate

Field Name	Description and Possible Values
Name	Name of the certificate.
Cert Type	Select OTHER for NF certificates.
Network Function	Name of the NF.
Purpose	Purpose of the NF certificate.
Issuer	Name of the issuer for the certificate.
Life Cycle Management	Possible values are MANUAL and AUTOMATIC.

5. Under **Private Key Options**, enter the following information:

Table 3-16 Private Key Options

Field Name	Possible Values
Key Algorithm	RSA, EC
Key Encoding	DER, PEM
Key Size	KEYSIZE_2048, KEYSIZE_4096
Elliptic Curve	SECP256r1, SECP384r1

Figure 3-33 Private Key Options

6. Under **Private Key Output**, enter the following information:

Table 3-17 Private Key Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Kubernetes Secret Name.
Key	Kubernetes secret key against which the key-pair will be stored.

Figure 3-34 Private Key Output

7. Under **Public Key Certificate Options**, enter the following:

Table 3-18 Public Key Certificate Options

Field Name	Description
Key Usage	Value(s): DIGITAL_SIGNATURE
Extended Key Usage	Value(s): CLIENT_AUTH and SERVER_AUTH
Basic Constraints	Value(s): END_ENTITY
Subject	Country: Enter country code: State: Enter state code. Location: City or town where company is legally located. Organization: Name of your organization. Organisation Unit: Name of business unit. Common Name: The Common Name (CN) represents the server name to be protected by the certificate. Requested Validity (Days): Number of days requested for which the certificate will be valid.

Figure 3-35 Public Key Certificate Options

Public Key Certificate Options

Key Usage

Critical: ☒

Value(s):

Extended Key Usage

Critical: ☒

Value(s):

Basic Constraints

Critical: ☒

Value:

Subject

Country:

State:

Location:

8. Under **Subject Alternate Names**, enter the following:

Table 3-19 Subject Alternate Names

Field Name	Description
IP Address	The IPs you want to protect under this certificate.
DNS Names	List of DNS domain names.
URI ID API Roots	List of URI ID (API root of the NF Instance).
URI ID URNs	List of URI ID (URN of the NFInstanceId).

Figure 3-36 Subject Alternate Names

Subject Alternate Names

Critical:

☒

IP Addresses:

IP Addresses

DNS Names:

DNS Names

URI ID API Roots:

URI ID API Roots

URI ID URNs:

URI ID URNs

9. Under Certificate Output, enter the following for the NF certificate:

Table 3-20 Certificate Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Name of the secret.
Key	The key against which the certificate will be populated.

Figure 3-37 Certificate Output

Certificate Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

10. (Optional) Under Certificate Chain Output, enter the following:

Table 3-21 Certificate Chain Output

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Name of the secret.
Key	Kubernetes secret key against which the certificate chain will be stored.

Figure 3-38 Certificate Chain Output

Certificate Chain Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

If the **Certificate Chain Output** section is filled, then the certificate chain can either be obtained from the CA or can be configured manually. This is based on the

`extractCertChainFromCmpResponse` helm parameter. For more information, see *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide*

Note

`extractCertChainFromCmpResponse`: This field when set true specifies that certificate chain will be extracted from CA's CMP response message. When false, the operator can configure the chain manually. This certificate chain is used in the TLS handshake along with the certificate.

11. (Optional) under **CA Bundle Input**, enter the following information:

Table 3-22 CA Bundle Input

Field Name	Description
Namespace	Name of the namespace.
Secret Name	Name of the secret.
Key	Kubernetes secret key against which CA bundle certificate(s) will be stored.

Figure 3-39 CA Bundle Input

The screenshot shows a form titled "CA Bundle Input" with three input fields: "Namespace:", "Secret Name:", and "Key:". Each field has a corresponding text input box to its right.

12. Click **Save**.

For sample NF configuration, see [Creating NF Certificate Using OCCM - Sample Configuration](#).

3.5.3 Renew NF Certificates

To renew NF certificates:

- OCCM sends CMPv2 Key Update Request (KUR) to the CA.
- KUR is used to renew OCCM certificate (CMP Identity Key) and NF Certificates.
- The KUR can be signed either by the OCCM key and certificate or by the certificate that is being renewed and its corresponding key. The corresponding certificate is included in `extraCerts`.

To renew certificates:

1. **Set the Key Update Request mode:**

Certificate renewal is a CMP KUR procedure. You can configure OCCM to sign the KUR in two ways:

- Using OCCM key and certificate.
- Using the certificate that is being renewed and its corresponding key.

You can use the `occmConfig.cmp.config.useKurOldCertMode` parameter to determine how OCCM will sign the KUR at the time of deployment.

- If `occmConfig.cmp.config.useKurOldCertMode` is set to true, OCCM key and certificate will be used to sign the CMP KUR message.
- If `occmConfig.cmp.config.useKurOldCertMode` is set to false, the certificate that is being renewed will be used.

By default, this parameter is set to false.

2. Configure Renew Before Expiration (days):

OCCM monitors the certificate validity and initiates automatic certificate renewal based on the renew before period configuration. You can update the **Renew Before Expiration (Days)** field on the **Create Certificate** page at the time of certificate creation. This field specifies the number of days before the certificate expiry date when the certificate must be renewed.

Figure 3-40 Renew Before Expiration (Days)

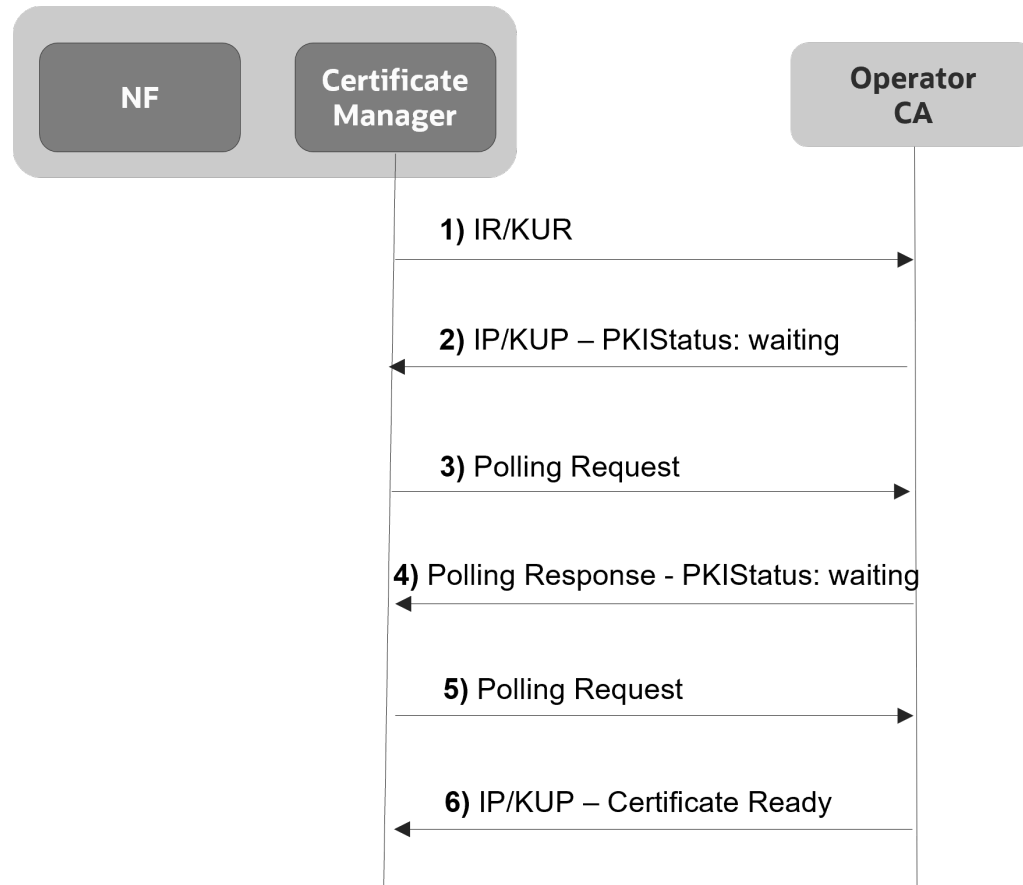
The screenshot shows the 'Create Certificate' form with the following fields and values:

Field Label	Value
UUID:	UUID
Name:	Name
Cert Type:	OCCM
Network Function:	Network Function
Purpose:	Purpose
Issuer:	Issuer
Life Cycle Management:	MANUAL
Delete Secret on Cert Delete:	☒
Scope:	Scope
Renew Before Expiration (Days):	14

3.5.4 Polling for Certificates

After the IR or KUR, if the certificate is not available yet, the CA responds with PKI status 'Waiting'. The application keeps polling until the CA is ready with the certificate. Openssl implicitly handles polling. No additional configuration is required at the application level in this regard. However, the Total Timeout field can be set in the issuer configuration, which can restrict this polling time. It is the maximum total number of seconds a transaction may take, including polling etc. If the time specified by total timeout has elapsed, the polling will stop.

Figure 3-41 Polling for Certificates



3.5.5 Deleting the Certificate Configuration

To delete the certificate configuration:

1. Log in to CNC Console using your login credentials and select the **OCCM** Instance.
2. Click **OCCM** from the left pane and then click **Certificate**.
3. Click **Delete** and click **OK** on the confirmation prompt to delete the certificate.

Note

This procedure only deletes the certificate configuration from OCCM.
Run the following command to delete the Kubernetes secret holding the certificates:

```
kubectl delete secrets <secret name> -n <namespace>
```

For example:

```
kubectl delete secrets nrf-tls-secret -n ns1
```

3.6 OCCM Retry on Failure

OCCM supports retry on encountering failures during the certificate creation, certificate renewal and manipulation of Kubernetes secrets.

- The procedure is retried until successful or interrupted by an action executed by the operator.
- Retry is not controlled through any maximum limit.
- The retry interval is a pre-defined value and set to 30s.

Some of the failure scenarios for which retries will be attempted:

- CA is unavailable, not reachable, or busy.
- Any errors returned by CA.

OCCM also provides a retry mechanism for errors encountered during Kubernetes secret update with the generated key and certificate. Based on the error encountered (insufficient permissions, Kubernetes internal errors etc), once the User fixes the issue, the Kubernetes secrets are automatically updated due to the ongoing retries.

Note

In this case, there is no attempt to recreate the key and certificate. The retry is restricted to updating the Kubernetes secrets with the key and certificate that are already generated.

3.7 Network Policies

Network Policies are an application-centric construct that allow you to specify how a pod communicates with various network entities. It creates pod-level rules to control communication between the cluster pods and services, and to determine which pods and services can access one another inside a cluster.

Previously, the pods under deployment could be contacted by any other pods in the Kubernetes cluster without any restrictions. Now, Network Policy provides namespace-level isolation, which allows secured communications to and from OCCM with rules defined in the respective Network Policies. The network policies enforce access restrictions for all the applicable data flows except communication from Kubernetes node to pod for invoking container probe. For example, OCCM internal microservices can't be contacted directly by any other pods.

Managing Support for Network Policies

Enable

To use this feature, network policies need to be applied to the namespace where OCCM is applied.

Configure

You can configure this feature using Helm. For information about Configuring Network Policy for OCCM deployment, see *Oracle Communications Certificate Management Installation, Upgrade, and Fault Recovery Guide*.

Observe

There are no specific metrics and alerts required for the Support of Network Policy functionality.

4

Introducing OCCM in an Existing NF Deployment

This section describes the procedure to introduce OCCM in an existing NF deployment where certificates are managed manually. OCCM helps in automating certificate management.

You can move from manual management to automated manages in one of 2 ways:

- Using existing key and certificate.
- Using a new key and certificate.

Moving NFs from Manual Certificate Management to Automated Certificate Management with Existing Key and Certificate

To move NFs from manual certificate management to automated certificate management with existing key and certificate:

1. Configure a key and certificate on OCCM. You must reuse the same Kubernetes secret and the content as used by NF with manually generated key and certificate. The NF configuration must not be updated.
2. OCCM monitors the existing key and certificate in the configured Kubernetes secret and renews it. The metrics attached to the key and certificate are generated.

Note

The existing key and certificate are not validated against the configuration. However, the renewed certificate will be aligned with the configuration.

Moving NFs from Manual Certificate Management to Automated Certificate Management With new Key and Certificate

To move NFs from manual certificate management to automated certificate management with new key and certificate:

1. Configure a key and certificate on OCCM making sure to reuse the same Kubernetes secret as used by NF with manually generated key and certificate. Reusing the Kubernetes secret make sure that the NF configuration is not updated.
2. OCCM creates a new key and certificate in the configured Kubernetes secret and deletes the old key and certificate. The old key and certificate is deleted to generate OCCM metrics attached to the certificate creation.

Procedure

The operator can select the following values for the LCM Type field:

- Manual (With existing key and certificate)
- Automatic (With new key and certificate)

Manual

- In Manual mode, existing certificates are configured at OCCM so that OCCM can manage the lifecycle of certificates. For example, the certificates that are already being used by NFs can be monitored by OCCM and further renewed by OCCM. In this case, the same Kubernetes secret and the content as used by NF with manually generated key and certificate is reused by OCCM.

Note

The existing key and certificate are not validated against the configuration. Renewed certificate will be aligned with the configuration though.

Automatic

- In Automatic mode, OCCM can create fresh certificates, or override the existing certificate with a new one. For example, if NFs want to create a new key and certificate to override old one through OCCM, and monitor them, then a key and certificate can be created on OCCM using the same Kubernetes secret as used by NF with manually generated key and certificate
- OCCM creates a new key and certificate in the configured Kubernetes secret and deletes the old key and certificate

Note

While reusing the Kubernetes secret and content, you must ensure that the NF configuration is not updated.

Table 4-1 Dependency of LCM type on Kubernetes Secret

LCM Type	Description
Manual	Operator doesn't need to create a new secret. OCCM uses the existing Kubernetes secret.
Automatic	Operator can either create a new secret or use the existing Kubernetes secret with the Override Secret flag

Table 4-2 Behaviour of different LCM Types

LCM Type	Preexisting Kubernetes Secret	Override Secret Flag	Behaviour
Automatic	No	No Impact	Certificate is created irrespective of the override flag.

Table 4-2 (Cont.) Behaviour of different LCM Types

LCM Type	Preexisting Kubernetes Secret	Override Secret Flag	Behaviour
Automatic	Yes	True or False	True: The Kubernetes secret is overridden. False: An error is thrown because you must either use a new secret or set the override flag to true. This error is thrown upfront on the user interface or in the response if APIs are used.
Manual	No	NA	An error is thrown because OCCM expects a preconfigured Kubernetes secret. This error is thrown upfront on the user interface or in the response if APIs are used.
Manual	Yes	NA	Certificate configuration is created at OCCM for further certificate renewal and monitoring.

Moving Back to Manual Certificate Management

- If the operator wants to move back to manual certificate monitoring, then they can delete the entry from the OCCM configuration. OCCM doesn't delete the secret when the entry is deleted and the certificate can be monitored manually (if operator used same secret location).
- If user creates a separate secret during certificate management from OCCM, and the operator doesn't want to use the secret further, then operator can delete the entry from OCCM and must also delete the Kubernetes secret.

5

Accessing OCCM Resources Through Curl and Postman

CNC Console provides a secure option for accessing OCCM resources through curl and postman using the CNC Console IAM access token. This section describes how to generate access tokens and access OCCM APIs.

5.1 Generate Access Tokens

CNC Console IAM provides a REST API for generating and refreshing access tokens.

To generate access tokens:

1. Send a POST request to the following URL to get an access token from CNC Console IAM:
`http://${cncc-iam-ingress-extrenal-ip}:${cncc-iam-ingress-service-port}/cncc/auth/realms/${realm}/protocol/openid-connect/token`

For example: `https://{host}:{port}/cncc/auth/realms/cncc/protocol/openid-connect/token`

2. The body of the request must be *x-www-form-urlencoded* encoded as follows:

```
'client_id': 'your_client_id',  
'username': 'your_username',  
'password': 'your_password',  
'grant_type': 'password'
```

For example:

```
'client_id': 'cncc-api-access',  
'username': 'user1',  
'password': '*****',  
'grant_type': 'password'
```

3. Run the following curl command to generate access tokens:

```
curl --location --request POST 'http://{host}:{port}/cncc/auth/realms/cncc/protocol/openid-connect/token' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'grant_type=password' \  
--data-urlencode 'username=user1' \  
--data-urlencode 'password=*****' \  
--data-urlencode 'client_id=cncc-api-access'
```

4. In response, you will get an **access_token** and a **refresh_token**:

```
{  
  "access_token": "eyJhbGc...09l2Q",
```

```

    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhbG...5vKPF-ZIg",
    "token_type": "bearer",
    "not-before-policy": 0,
    "session_state": "6c42d978-14ac-4793-a1e3-789cfbdb2b74",
    "scope": "email profile"
  }

```

5.2 Refresh Access Tokens

If the access token has expired, you can refresh it by sending a POST request to the same URL, but containing the refresh token instead of username and password:

Perform the following procedure to refresh the access tokens:

If the `access_token` has expired, it can be refreshed by sending a POST request to the same URL as above; but the POST method must have the refresh token instead of username and password. The format is as follows:

```

'client_id': 'your_client_id',
'refresh_token': refresh_token_from_previous_request,
'grant_type': 'refresh_token'

```

For Example:

```

'client_id': 'cncc-api-access',
'refresh_token': 'eyJhbGciOiJIU...dKnMfb5vKPF-ZIg',
'grant_type': 'refresh_token'

```

In response, you will receive a new **access_token** and **refresh_token**.

5.3 Issuer Configuration API Access

You need the CNC Console IAM access tokens to access the OCCM Issuer APIs through CNC Console.

You must include the following headers when you send an API request:

- **Authorization:** The access token must be used in every request to a NF resource by placing it in the *Authorization* header.
- **oc-cncc-id:** M-CNCC uses the `oc-cncc-id` header to find the agent or master owning the instance.
- **oc-cncc-instance-id:** A-CNCC Core (or M-CNCC Core) uses the `oc-cncc-instance-id` header to find the NF instance for routing.

Following headers must be passed in the curl or postman request while accessing the OCCM Issuers resource:

HTTP Request:

```

curl --request POST 'http://${occm-external-ip}:${occm-service-port}/occm-
config/v1/issuers/'
--header 'Content-Type: application/json'

```

```

--header 'oc-cncc-id: Cluster1'
--header 'oc-cncc-instance-id: Cluster1-occm-instance1'
--header 'Authorization: Bearer <Token>'
--data-raw '{
  "name": "CA1",
  "server": "http://ca1-openssl-mock.ns1.svc.thrust5:8090",
  "recipientDN": "/CN=x.company.com",
  "issuerDN": "/CN=x.company.com",
  "totalTimeout": "720",
  "messageTimeout": "120",
  "cmpProtectionOccmCert": {
    "type": null,
    "digestAlgorithm": null,
    "macAlgorithm": null,
    "macK8sSecretIn": {
      "namespace": "",
      "name": "",
      "passKey": "",
      "refKey": ""
    },
    "signK8sSecretIn": {
      "namespace": "",
      "name": "",
      "key": "",
      "cert": "",
      "extraCerts": []
    }
  },
  "cmpProtectionOtherCert": {
    "type": "SIGNATURE",
    "digestAlgorithm": "SHA256",
    "signK8sSecretIn": {
      "namespace": "ns1",
      "name": "ca1-cmp-identity-secret",
      "key": "cmpkey.pem",
      "cert": "cmpcert.pem",
      "extraCerts": []
    }
  },
  "occmTrustStoreK8sSecretIn": {
    "namespace": "ns1",
    "name": "ca1-occm-trust-store-secret",
    "rootCACerts": [
      "caroot.pem"
    ],
    "intCACerts": [
      "intcacert.pem"
    ],
    "serverCert": "servercert.pem"
  },
  "tlsConfig": {
    "enableTLS": false,
    "tlsTrustStoreK8sSecretItem": {
      "namespace": "",
      "name": "",

```

```

        "tlsTrustedCerts": []
    }
}
}'

```

HTTPS Request

```

curl --request POST 'http://${occm-external-ip}:${occm-service-port}/occm-
config/v1/issuers/'
--header 'Content-Type: application/json'
--header 'oc-cncc-id: Cluster1'
--header 'oc-cncc-instance-id: Cluster1-occm-instance1'
--header 'Authorization: Bearer <Token>'
--data-raw '{
  "name": "CA1",
  "server": "https://ca1-openssl-mock.ns1.svc.thrust5:8443",
  "recipientDN": "/CN=x.company.com",
  "issuerDN": "/CN=x.company.com",
  "totalTimeout": "720",
  "messageTimeout": "120",
  "cmpProtectionOcmCert": {
    "type": null,
    "digestAlgorithm": null,
    "macAlgorithm": null,
    "macK8sSecretIn": {
      "namespace": "",
      "name": "",
      "passKey": "",
      "refKey": ""
    },
    "signK8sSecretIn": {
      "namespace": "",
      "name": "",
      "key": "",
      "cert": "",
      "extraCerts": []
    }
  },
  "cmpProtectionOtherCert": {
    "type": "SIGNATURE",
    "digestAlgorithm": "SHA256",
    "signK8sSecretIn": {
      "namespace": "ns1",
      "name": "ca1-cmp-identity-secret",
      "key": "cmpkey.pem",
      "cert": "cmpcert.pem",
      "extraCerts": []
    }
  },
  "occmTrustStoreK8sSecretIn": {
    "namespace": "ns1",
    "name": "ca1-occm-trust-store-secret",
    "rootCACerts": [
      "caroot.pem"
    ]
  },
}
```

```

        "intCACerts": [
            "intcacert.pem"
        ],
        "serverCert": "servercert.pem"
    },
    "tlsConfig": {
        "enableTLS": true,
        "tlsTrustStoreK8sSecretItem": {
            "namespace": "ns1",
            "name": "ca1-tls-trust-store-secret",
            "tlsTrustedCerts": ["caroot.cer"]
        }
    }
}
}'

```

5.4 Certificate Configuration API Access

You need the CNC Console IAM access token that you generated to access OCCM Certificates Configuration API:

```

curl --request POST 'http://${occm-external-ip}:${occm-service-port}/occm-
config/v1/certs/'
--header 'Content-Type: application/json'
--header 'oc-cncc-id: Cluster1'
--header 'oc-cncc-instance-id: Cluster1-occm-instance1'
--header 'Authorization: Bearer <Token>'
--data-raw ' {

    "name": "NRF TLS Cert",
    "lcmType": "AUTOMATIC",
    "certType": "OTHER",
    "renewBefore": "14",
    "certPurpose": "NRF SBI",
    "issuer": "CA1",
    "privateKey": {
        "keyAlgo": "RSA",
        "keySize": "KEYSIZE_2048",
        "keyEncoding": "PEM",
        "ecCurve": null,
        "privateKeyK8sSecretOut": {
            "namespace": "ns1",
            "name": "nrf-tls-secret",
            "key": "nrfkey.pem"
        }
    },
    "csr": {
        "extendedKeyUsage": {
            "critical": false,
            "extendedKeyUsageValues": [
                "CLIENT_AUTH",
                "SERVER_AUTH"
            ]
        },
        "keyUsage": {

```

```

        "critical": true,
        "keyUsageValues": [
            "DIGITAL_SIGNATURE"
        ]
    },
    "basicConstraints": {
        "critical": false,
        "basicConstraintsValue": "END_ENTITY"
    },
    "subject": {
        "country": "IN",
        "state": "KA",
        "location": "BLR",
        "organization": "Oracle",
        "organizationUnit": "CGBU",
        "commonName": "a.company.com"
    },
    "days": "365",
    "subjectAltName": {
        "critical": false,
        "ipAddress": [
            "10.10.10.20",
            "10.10.10.21"
        ],
        "dns": [
            "y.company.com",
            "z.company.com"
        ],
        "uriIdApiRoot": null,
        "uriIdUrn": [
            "urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
        ]
    },
    "certK8sSecretOut": {
        "namespace": "ns1",
        "name": "nrf-tls-secret",
        "key": "nrfcert.pem"
    },
    "certChainK8sSecretOut": {
        "namespace": "ns1",
        "name": "nrf-tls-secret",
        "key": "nrfcertchain.pem"
    }
},
"caBundleK8sSecretIn": {
    "namespace": "ns1",
    "name": "ca-bundle-secret",
    "key": "ca-bundle.pem"
},
"nf": "NRF",
"overrideSecret": false
}'

```

5.5 Logging API Access

You need the CNC Console IAM access token that you generated to access OCCM Logging APIs.

```
curl --location --request PUT 'http://host:port/occm-config/v1/occm/logging' \
--header 'oc-cncc-id: Cluster1' \
--header 'oc-cncc-instance-id: Cluster1-occm-instance1' \
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiIs...' \
--header 'Content-Type: application/json' \
--data-raw '{
  "appLogLevel": "DEBUG",
  "packageLogLevel": [
    {
      "packageName": "root",
      "logLevelForPackage": "ERROR"
    }
  ]
}
```

6

OCCM Metrics

This chapter provides information about metrics for OCCM.

Dimension Description

The following table describes different types of metric dimensions:

Table 6-1 OCCM Dimension Description

Dimension	Description	Possible Values
method	Http method	GET, PUT, POST, DELETE
httpVersion	Http protocol version	HTTP/1.1
scheme	Http protocol scheme	HTTP, UNKNOWN
uri	URL of requested API	/occm-config/v1/certs
nfType	API called by NF	eg: SCP, NRF, OCCM
statusCode	Http status code	200, 202
certUuid	Unique ID for the purpose of logging and tracking	eg: 7523a545-089b-49e9-a05c-ae5141db544b
requestType	Type of request	IR, KUR
certName	Name of the certificate	NRFTLS-1, SCPTLS-1
certPurpose	Purpose of the certificate creation	NRF SBI
issuerName	Name of the Issuer	CA
errorReason	Reason of the error	eg:ERR_K8S_SECRET_CREATION_ERROR
operationType	Type of operation	CREATE, RENEW, DELETE
host	Application hosted on cluster	eg: occm.occncc-thrust5-01.svc.thrust5
application	Name of the application	OCCM
caServer	URL of the Certificate Authority (Issuer)	eg: http://ca1-openssl-mock.occncc-thrust5-01.svc.thrust5:8089 , https://ca2-openssl-mock.occncc-thrust5-01.svc.thrust5:8443
status	To know the status of openssl CMP cmd	SUCCESS, FAILED

6.1 occm_config_http_requests_total

Table 6-2 occm_config_http_requests_total

Field	Details
Description	OCCM Configuration total HTTP request counter metric
Type	Counter

Table 6-2 (Cont.) occm_config_http_requests_total

Field	Details
Dimensions	- host - application - httpVersion - scheme - method - nfType - uri
Example	<pre>occm_config_http_requests_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", host="occm.occncc-thrust5-01.svc.thrust5", httpVersion="HTTP/1.1", instance="10.233.121.228:9000", job="occne-infra/occne-nf-cnc- podmonitor", method="POST", namespace="occncc-thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8", scheme="http", uri="/occm-config/v1/certs"}</pre>

6.2 occm_config_http_response_total

Table 6-3 occm_config_http_response_total

Field	Details
Description	OCCM Configuration total HTTP response counter metric
Type	Counter
Dimensions	- host - httpVersion - scheme - method - nfType - statusCode - uri
Example	<pre>occm_config_http_responses_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", host="occm.occncc-thrust5-01.svc.thrust5", httpVersion="HTTP/1.1", instance="10.233.121.228:9000", job="occne-infra/occne-nf-cnc- podmonitor", method="POST", namespace="occncc-thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8", scheme="http", statusCode="202", uri="/occm-config/v1/certs"}</pre>

6.3 occm_cmp_requests_total

Table 6-4 occm_cmp_requests_total

Field	Details
Description	OCCM total CMP request counter metric
Type	Counter
Dimensions	- certUid - certName - nfType - requestType - issuerName - caServer
Example	<pre>occm_cmp_requests_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", caServer="http://ca90-openssl-mock.occncc- thrust5-01.svc.thrust5:8083", certName="NRFTLS-47", certUid="c0578b02-caab-454a- bd97-422b0e1c575b", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", issuerName="CA90", job="occne-infra/occne-nf-cnc-podmonitor", namespace="occncc-thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8", requestType="ir"}</pre> <pre>occm_cmp_requests_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", caServer="http://ca90-openssl-mock.occncc- thrust5-01.svc.thrust5:8083", certName="NRFTLS-47", certUid="c0578b02-caab-454a- bd97-422b0e1c575b", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", issuerName="CA90", job="occne-infra/occne-nf-cnc-podmonitor", namespace="occncc-thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8", requestType="kur"}</pre>

6.4 occm_cmp_responses_total

Table 6-5 occm_cmp_responses_total

Field	Details
Description	OCCM total CMP response counter metric
Type	Counter
Service Operation	

Table 6-5 (Cont.) occm_cmp_responses_total

Field	Details
Dimensions	- certUid - certName - nfType - requestType - status - statusCode - issuerName - caServer
Example	<pre>occm_config_http_responses_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", host="occm.occncc-thrust5-01.svc.thrust5", httpVersion="HTTP/1.1", instance="10.233.121.228:9000", job="occne-infra/occne-nf-cnc- podmonitor", method="POST", namespace="occncc-thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8", scheme="http", statusCode="202", uri="/occm-config/v1/certs"}</pre>

6.5 occm_cert_expiry

Table 6-6 occm_cert_expiry

Field	Details
Description	OCCM Cert expiry gauge metrics. It will indicate Certificate expiry timestamp.
Type	Gauge
Dimensions	- certUid - certName - nfType - issuerName - certPurpose
Example	<pre>occm_cert_expiry{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", certName="NRF-TLS-47", certPurpose="NRF SBI", certUid="c0578b02-caab-454a-bd97-422b0e1c575b", container="occm", endpoint="cnc- metrics", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", issuerName="CA90", job="occne-infra/occne-nf-cnc-podmonitor", namespace="occncc- thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8"}</pre>

6.6 occm_cert_status

Table 6-7 occm_cert_status

Field	Details
Description	OCCM Cert status gauge metric. It will indicate Certificate status CREATING(1), READY(2), FAILED(3), DELETED(6), EXPIRED(7), WAITING(8)
Type	Gauge
Dimensions	- certUid - nfType - certName - certPurpose - issuerName
Example	<pre>occm_cert_status{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", certName="NRFTLS-47", certPurpose="NRF SBI", certUid="c0578b02-caab-454a-bd97-422b0e1c575b", container="occm", endpoint="cnc- metrics", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", issuerName="CA90", job="occne-infra/occne-nf-cnc-podmonitor", namespace="occncc- thrust5-01", nfType="NRF", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8"}</pre>

6.7 occm_cmp_cli_durations

Table 6-8 occm_cmp_cli_durations

Field	Details
Description	OCCM cmp cli duration histogram metrics . CMP cli time taken in between request and response from CA
Type	Histogram
Dimensions	- certUid - nfType - certName - requestType - caServer
Example	<pre>occm_cmp_cli_durations_bucket{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", caServer="http://ca90-openssl-mock.occncc- thrust5-01.svc.thrust5:8083", certName="NRFTLS-47", certUid="c0578b02-caab-454a- bd97-422b0e1c575b", container="occm", endpoint="cnc-metrics", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", job="occne-infra/occne-nf- cnc-podmonitor", le="5.0", namespace="occncc-thrust5-01", nfType="NRF", pod="occm- occm-67764765f8-7rpm8", pod_template_hash="67764765f8", requestType="kur"}</pre>

6.8 occm_cert_request_status_total

Table 6-9 occm_cert_request_status

Field	Details
Description	OCCM Certificate request status counter metric. It will indicate certificate status, error reason, operation type whether CREATE or Renew etc.
Type	Counter
Dimensions	- certName - certUuid - errorReason - issuerName - nfType - operationType
Example	<pre>occm_cert_request_status_total{app_kubernetes_io_application="occm", app_kubernetes_io_component="occm", app_kubernetes_io_engVersion="24.1.0", app_kubernetes_io_instance="occm", app_kubernetes_io_managed_by="Helm", app_kubernetes_io_microservice="occm", app_kubernetes_io_mktgVersion="24.1.0.0.0", app_kubernetes_io_name="occm", app_kubernetes_io_part_of="occm", app_kubernetes_io_vendor="Oracle", app_kubernetes_io_version="24.1.0.0.0", application="occm", certName="NRFTLS-47", certUuid="c0578b02-caab-454a- bd97-422b0e1c575b", container="occm", endpoint="cnc-metrics", errorReason="OK", helm_sh_chart="occm-24.1.0", instance="10.233.121.228:9000", issuerName="CA90", job="occne-infra/occne-nf-cnc-podmonitor", namespace="occncc-thrust5-01", nfType="NRF", operationType="RENEW", pod="occm-occm-67764765f8-7rpm8", pod_template_hash="67764765f8"}</pre>

7

OCCM Alerts

This section describes the alerts available for OCCM.

Note

Alert file is packaged with OCCM CSAR package.

- Review the `occm_alerting_rules_promha_<version>.yaml` file and edit the value of the parameters in the `occm_alerting_rules_promha_<version>.yaml` file (if needed to be changed from default values) before configuring the alerts. See above table for details.
- `kubernetes_namespace` is configured as `kubernetes` namespace in which OCCM is deployed. Default value is `occm`. Please update the `occm_alerting_rules_promha_<version>.yaml` file to reflect the correct OCCM `kubernetes` namespace.

7.1 OccmCertExpiryWithinMinorThreshold

Table 7-1 OccmCertExpiryWithinMinorThreshold

Field	Details
Description	OCCM Certificate Expiry Alert The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> for <code>{{ \$labels.certPurpose }}</code> will expire soon within 90 days
Summary	namespace: <code>{{ \$labels.namespace }}</code> , podname: <code>{{ \$labels.pod }}</code> , timestamp: <code>{{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}</code> : The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> for <code>{{ \$labels.certPurpose }}</code> will expire soon within 90 days
Severity	Minor
Condition	Certificate will expire soon within 90 days
OID	1.3.6.1.4.1.323.5.3.54.1.2.7001
Metric Used	<code>occm_cert_expiry</code>
Recommended Actions	Information that Certificate is going to expire within 90 days. The alert is cleared when the certificate is renewed so that the Certificate expiry days falls below the Minor threshold or when the Certificate expiry days crosses the Major threshold, in which case the <code>OccmCertExpiryWithinMajorThreshold</code> alert is raised. Note: The threshold is configurable in the <code>occm_alertingrules_<version>.yaml</code> file. Steps: 1. Check certificate configuration for renew before days. 2. If this is unexpected, contact My Oracle Support.

7.2 OccmCertExpiryWithinMajorThreshold

Table 7-2 OccmCertExpiryWithinMajorThreshold

Field	Details
Description	OCCM Certificate Expiry Alert The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} will expire soon within 30 days
Summary	namespace: {{labels.namespace}}, podname: {{labels.pod}}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} will expire soon within 30 days
Severity	Major
Condition	Certificate will expire soon within 30 days
OID	1.3.6.1.4.1.323.5.3.54.1.2.7001
Metric Used	occm_cert_expiry
Recommended Actions	Information that Certificate is going to expire within 30 days. The alert is cleared when the certificate is renewed or when the Certificate expiry days crosses the Critical threshold, in which case OccmCertExpiryWithinCriticalThreshold alert is raised. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Check certificate configuration for renew before days. 2. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 3. Depending on the failure reason, take the resolution steps. 4. If this is unexpected, contact My Oracle Support.

7.3 OccmCertExpiryWithinCriticalThreshold

Table 7-3 OccmCertExpiryWithinCriticalThreshold

Field	Details
Description	OCCM Certificate Expiry Alert The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} will expire soon within 1 week
Summary	namespace: {{labels.namespace}}, podname: {{labels.pod}}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} will expire soon within 1 week
Severity	Critical
Condition	Certificate will expire soon within 1 week
OID	1.3.6.1.4.1.323.5.3.54.1.2.7001

Table 7-3 (Cont.) OccmCertExpiryWithinCriticalThreshold

Field	Details
Metric Used	occm_cert_expiry
Recommended Actions	Information that Certificate is going to expire within 1 week. The alert is cleared when the certificate is renewed. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Check certificate configuration for renew before days. 2. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 3. Depending on the failure reason, take the resolution steps. 4. If this is unexpected, contact My Oracle Support.

7.4 OccmCertExpired

Table 7-4 OccmCertExpired

Field	Details
Description	OCCM Certificate has Expired Critical Alert The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} is expired
Summary	namespace: {{labels.namespace}}, podname: {{labels.pod}}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{labels.certName}} used by {{labels.nfType}} for {{labels.certPurpose}} is expired
Severity	Critical
Condition	Certificate has expired
OID	1.3.6.1.4.1.323.5.3.54.1.2.7002
Metric Used	occm_cert_expiry
Recommended Actions	Information that Certificate has expired. The alert is cleared when the certificate is renewed. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. Refer user guide for procedure to renew expired certificate. 4. If this is unexpected, contact My Oracle Support.

7.5 OccmServiceDown

Table 7-5 OccmServiceDown

Field	Details
Description	OCCM Service Down Alert New certificates will not be created, and existing ones can not be renewed until OCCM is back
Summary	namespace: {{\$labels.namespace}}, podname: {{\$labels.pod}}, timestamp: {{ with query "time()" }}{ . first value humanizeTimestamp }}{ end }}: OCCM service is down
Severity	Critical
Condition	The pods of the occm service is unavailable.
OID	1.3.6.1.4.1.323.5.3.54.1.2.7004
Metric Used	up Note: This is a prometheus metric used for instance availability monitoring. If this metric is not available, use the similar metric as exposed by the monitoring system.
Recommended Actions	The alert is cleared when the occm service is available. Steps: 1. Check the orchestration logs of occm service and check for liveness or readiness probe failures. 2. Refer to the application logs on Kibana and filter based on occm service names. Check for ERROR WARNING logs related to thread exceptions. 3. Depending on the failure reason, take the resolution steps. 4. In case the issue persists, contact My Oracle Support.

7.6 OccmMemoryUsageMinorThreshold

Table 7-6 OccmMemoryUsageMinorThreshold

Field	Details
Description	OCCM Memory Usage Alert OCCM Memory Usage for pod {{ \$labels.pod }} has crossed the configured minor threshold (70%) (value={{ \$value }}) of its limit.
Summary	namespace: {{\$labels.namespace}}, podname: {{\$labels.pod}}, timestamp: {{ with query "time()" }}{ . first value humanizeTimestamp }}{ end }}: Memory Usage of pod exceeded 70% of its limit.
Severity	Minor
Condition	A pod has reached the configured minor threshold(70%) of its memory resource limits.
OID	1.3.6.1.4.1.323.5.3.54.1.2.7005

Table 7-6 (Cont.) OccmMemoryUsageMinorThreshold

Field	Details
Metric Used	container_memory_usage_bytes, Note : This is a kubernetes metric used for instance availability monitoring. If the metric is not available, use the similar metric as exposed by the monitoring system.
Recommended Actions	The alert gets cleared when the memory utilization falls below the Minor Threshold or crosses the major threshold, in which case OccmMemoryUsageMajorThreshold alert shall be raised. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.7 OccmMemoryUsageMajorThreshold

Table 7-7 OccmMemoryUsageMajorThreshold

Field	Details
Description	OCCM Memory Usage Alert OCCM Memory Usage for pod {{ \$labels.pod }} has crossed the configured major threshold (80%) (value={{ \$value }}) of its limit.
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: Memory Usage of pod exceeded 80% of its limit.
Severity	Major
Condition	A pod has reached the configured major threshold(80%) of its memory resource limits.
OID	1.3.6.1.4.1.323.5.3.54.1.2.7005
Metric Used	container_memory_usage_bytes, Note : This is a kubernetes metric used for instance availability monitoring.If the metric is not available, use the similar metric as exposed by the monitoring system.

Table 7-7 (Cont.) OccmMemoryUsageMajorThreshold

Field	Details
Recommended Actions	The alert gets cleared when the memory utilization falls below the Major Threshold or crosses the critical threshold, in which case OccmMemoryUsageMajorThreshold alert shall be raised Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.8 OccmMemoryUsageCriticalThreshold

Table 7-8 OccmMemoryUsageCriticalThreshold

Field	Details
Description	OCCM Memory Usage Alert OCCM Memory Usage for pod {{ \$labels.pod }} has crossed the configured critical threshold (90%) (value={{ \$value }}) of its limit..
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: Memory Usage of pod exceeded 90% of its limit.
Severity	Critical
Condition	A pod has reached the configured critical threshold (90%) of its memory resource limits
OID	1.3.6.1.4.1.323.5.3.54.1.2.7005
Metric Used	container_memory_usage_bytes, Note : This is a kubernetes metric used for instance availability monitoring.If the metric is not available, use the similar metric as exposed by the monitoring system.
Recommended Actions	The alert gets cleared when the memory utilization falls below the Critical Threshold.Note : The threshold is configurable in the alerts.yaml Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.9 OccmCPUUsageMinorThreshold

Table 7-9 OccmCPUUsageMinorThreshold

Field	Details
Description	OCCM CPU Usage Alert OCCM Pod {{ \$labels.pod }} has high CPU usage detected.
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: CPU usage is {{ \$value printf "%.2f" }} which is usage is above 70% (current value is: {{ \$value }})
Severity	Minor
Condition	CPU usage is above 70%
OID	1.3.6.1.4.1.323.5.3.54.1.2.7006
Metric Used	container_cpu_usage_seconds_total
Recommended Actions	Information regarding CPU usage If it is above 70% The alert gets cleared when the CPU usage falls below the Minor Threshold. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.10 OccmCMPFailureMinor

Table 7-10 OccmCMPFailureMinor

Field	Details
Description	OCCM CMP Command Execution Failure Alert The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while executing CMP cmd with {{ \$labels.statusCode }}.
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while executing CMP cmd with {{ \$labels.statusCode }}.
Severity	Minor
Condition	Certificate has failed while executing CMP cmds.
OID	1.3.6.1.4.1.323.5.3.54.1.2.7007
Metric Used	occn_cmp_responses_total

Table 7-10 (Cont.) OccmCMPFailureMinor

Field	Details
Recommended Actions	Information that the rate of certificate failure due to CMP command execution error has crossed the threshold. The alert is cleared when the rate of certificate failure due to CMP command execution error falls below the Minor threshold or when the error rate crosses the Major threshold, in which case the OccmCMPFailureMajor alert is raised. Note: The threshold is configurable in the <code>occm_alertingrules_<version>.yaml</code> file. Steps: 1. Refer to the application logs on Kibana and filter based on <code>occm</code> service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.11 OccmCMPFailureMajor

Table 7-11 OccmCMPFailureMajor

Field	Details
Description	OCCM CMP Command Execution Failure Alert The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> has failed while executing CMP cmd with <code>{{ \$labels.statusCode }}</code>.
Summary	namespace: <code>{{ \$labels.namespace }}</code>, podname: <code>{{ \$labels.pod }}</code>, timestamp: <code>{{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}</code>: The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> has failed while executing CMP cmd with <code>{{ \$labels.statusCode }}</code>.
Severity	Major
Condition	Certificate has failed while executing CMP cmds
OID	1.3.6.1.4.1.323.5.3.54.1.2.7007
Metric Used	<code>occm_cmp_responses_total</code>
Recommended Actions	Information that the rate of certificate failure due to CMP command execution error has crossed the threshold. The alert is cleared when the rate of certificate failure due to CMP command execution error falls below the Major threshold or when the error rate crosses the Critical threshold, in which case the OccmCMPFailureCritical alert is raised. Note: The threshold is configurable in the <code>occm_alertingrules_<version>.yaml</code> file. Steps: 1. Refer to the application logs on Kibana and filter based on <code>occm</code> service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.12 OccmCMPFailureCritical

Table 7-12 OccmCMPFailureCritical

Field	Details
Description	OCCM CMP Command Execution Failure Alert The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while executing CMP cmd with {{ \$labels.statusCode }}.
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while executing CMP cmd with {{ \$labels.statusCode }}.
Severity	Critical
Condition	Certificate has failed while executing CMP cmds
OID	1.3.6.1.4.1.323.5.3.54.1.2.7007
Metric Used	occm_cmp_responses_total
Recommended Actions	Information that the rate of certificate failure due to CMP command execution error has crossed the threshold. The alert is cleared when the rate of certificate failure due to CMP command execution error falls below the Critical threshold. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.13 OccmFailureMinor

Table 7-13 OccmFailureMinor

Field	Details
Description	OCCM Internal Failure Alert The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while creating cert with {{ \$labels.errorReason }}.
Summary	namespace: {{ \$labels.namespace }}, podname: {{ \$labels.pod }}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{ \$labels.certName }} used by {{ \$labels.nfType }} has failed while creating cert with {{ \$labels.errorReason }}.
Severity	Minor
Condition	Certificate has failed while creating
OID	1.3.6.1.4.1.323.5.3.54.1.2.7008
Metric Used	occm_cmp_responses_total

Table 7-13 (Cont.) OccmFailureMinor

Field	Details
Recommended Actions	Information that the rate of OCCM errors has crossed the threshold. The alert is cleared when the rate OCCM error falls below the Minor threshold or when the error rate crosses the Major threshold, in which case the OccmFailureMajor alert is raised. Note: The threshold is configurable in the <code>occm_alertingrules_<version>.yaml</code> file. Steps: 1. Refer to the application logs on Kibana and filter based on <code>occm</code> service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.14 OccmFailureMajor

Table 7-14 OccmFailureMajor

Field	Details
Description	OCCM Internal Failure Alert The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> has failed while creating cert with <code>{{ \$labels.errorReason }}</code>.
Summary	namespace: <code>{{ \$labels.namespace }}</code>, podname: <code>{{ \$labels.pod }}</code>, timestamp: <code>{{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}</code>: The certificate <code>{{ \$labels.certName }}</code> used by <code>{{ \$labels.nfType }}</code> has failed while creating cert with <code>{{ \$labels.errorReason }}</code>.
Severity	Major
Condition	Certificate has failed while creating
OID	1.3.6.1.4.1.323.5.3.54.1.2.7008
Metric Used	<code>occm_cmp_responses_total</code>
Recommended Actions	Information that the rate of OCCM errors has crossed the threshold. The alert is cleared when the rate OCCM error falls below the Major threshold or when the error rate crosses the Critical threshold, in which case the OccmFailureCritical alert is raised. Note: The threshold is configurable in the <code>occm_alertingrules_<version>.yaml</code> file. Steps: 1. Refer to the application logs on Kibana and filter based on <code>occm</code> service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.15 OccmFailureCritical

Table 7-15 OccmFailureCritical

Field	Details
Description	OCCM CMP Command Execution Failure Alert The certificate {{labels.certName}} used by {{labels.nfType}} has failed while creating cert with {{labels.errorReason}}.
Summary	namespace: {{labels.namespace}}, podname: {{labels.pod}}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{labels.certName}} used by {{labels.nfType}} has failed while creating cert with {{labels.errorReason}}.
Severity	critical
Condition	Certificate has failed while creating
OID	1.3.6.1.4.1.323.5.3.54.1.2.7008
Metric Used	occm_cmp_responses_total
Recommended Actions	Information that the rate of certificate failure due to CMP command execution error has crossed the threshold. The alert is cleared when the rate of certificate failure due to CMP command execution error falls below the Critical threshold. Note: The threshold is configurable in the occm_alertingrules_<version>.yaml file. Steps: 1. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 2. Depending on the failure reason, take the resolution steps. 3. If this is unexpected, contact My Oracle Support.

7.16 OccmRenewBeforValidityCritical

Table 7-16 OccmRenewBeforValidityCritical

Field	Details
Description	OCCM Renew Before is greater than Cert Validity Alert The certificate {{labels.certName}} used by {{labels.nfType}} has failed because renew before validity is greater than cert validity {{labels.errorReason}}.
Summary	namespace: {{labels.namespace}}, podname: {{labels.pod}}, timestamp: {{ with query "time()" }}{{ . first value humanizeTimestamp }}{{ end }}: The certificate {{labels.certName}} used by {{labels.nfType}} has failed because renew before validity is greater than cert validity {{labels.errorReason}}.
Severity	critical
Condition	Certificate has failed because renew before validity is greater than cert validity
OID	1.3.6.1.4.1.323.5.3.54.1.2.7009

Table 7-16 (Cont.) OccmRenewBeforValidityCritical

Field	Details
Metric Used	occm_cert_request_status_total
Recommended Actions	Information that the Certificate has failed because renew before validity is greater than cert validity. Steps: 1. Check certificate configuration for renew before days. 2. Also Check the validity requested for the Certificate and validity assigned by the CA. 3. Refer to the application logs on Kibana and filter based on occm service name. Check for ERROR WARNING logs related to thread exceptions. 4. Depending on the failure reason, take the resolution steps. 5. If this is unexpected, contact My Oracle Support.

7.17 OCCM Alert and MIB Configuration in Prometheus

CNE supporting Prometheus HA

This section describes the measurement based Alert rules configuration for OCCM in Prometheus. You must use the updated `occm_alerting_rules_promha_<version>.yaml` file.

Run the following command to create or update the PrometheusRule resource specified in the alert YAML file:

```
$ kubectl apply -f occm_alerting_rules_promha_<version>.yaml
```

Disabling Alerts

This section describes the procedure to disable the alerts in OCCM. To disable alerts:

1. Edit `occm_alerting_rules_promha_<version>.yaml` file to remove specific alert.
2. Remove complete content of the specific alert from the `occm_alerting_rules_promha_<version>.yaml` file.
For example, if you want to remove `OccmServiceDown` alert, remove the complete content:

```
## ALERT SAMPLE START##
- alert: OccmServiceDown
  annotations:
    description: 'New certificates will not be created, and existing
ones can not be renewed until OCCM is back'
    summary: 'namespace: {{$labels.namespace}}, podname:
{{$labels.pod}}, timestamp: {{ with query "time()" }}{{ . | first | value
| humanizeTimestamp }}{{ end }}: OCCM service is down'
  expr: absent(up{pod=~".*occm.*", namespace="occm-ns"}) or
(up{pod=~".*occm.*", namespace="occm-ns"}) == 0
  labels:
```

```
severity: critical
oid: "1.3.6.1.4.1.323.5.3.54.1.2.7004"
namespace: ' {{ $labels.namespace }} '
podname: ' {{$labels.pod}} '
## ALERT SAMPLE END##
```

3. Perform Alert configuration.

Validating Alerts

Configure and Validate Alerts in Prometheus Server. Refer to OCCM Alert Configuration for procedure to configure the alerts.

After configuring the alerts in Prometheus server, a user can verify that by following steps:

1. Open the Prometheus server from your browser using the <IP>:<Port>
2. Navigate to Status and then Rules
3. Search OCCM. OCCMAAlerts list is displayed.

① Note

If you are unable to see the alerts, it means that the alert file has not loaded in a format which the Prometheus server accepts. Modify the file and try again.

Configuring SNMP-Notifier

Configure the IP and port of the SNMP trap receiver in the SNMP Notifier using following procedure:

1. Execute the following command to edit the deployment:

```
kubectl edit deploy <snmp_notifier_deployment_name> -n <namespace>
```

Example:

```
$ kubectl edit deploy occne-snmp-notifier -n occne-infra
```

2. Edit the destination as follows:

```
--snmp.destination=<destination_ip>:<destination_port>
```

Example:

```
--snmp.destination=10.75.203.94:162
```

MIB Files for OCCM

here are two MIB files which are used to generate the traps. The user need to update these files along with the Alert file in order to fetch the traps in their environment.

- `occm_mib_tc_<version>.mib`: This is considered as OCCM top level mib file, where the Objects and their data types are defined

- `occm_mib_<version>.mib`: This file fetches the Objects from the top level mib file and based on the Alert notification, these objects can be selected for display.

Note

MIB files are packaged along with OCCM CSAR package. Download the file from MOS. For more information, see *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide*.

8

OCCM KPIs

This section describes the KPIs available for OCCM.

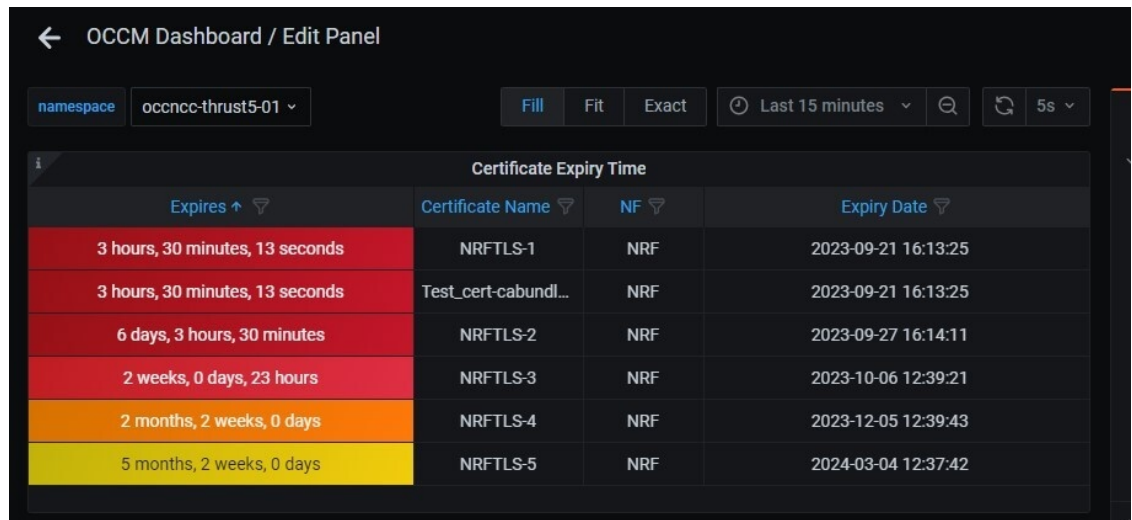
8.1 Certificate Expiry Time

Table 8-1 Certificate Expiry Time

Field	Details
Description	Certificate Expiry Time to list NF, Certificate Name, Expiry Date
Expression	OCCM dashboard in grafana will show Certificate Expiry Time panel with columns. Table visualization listing Expires, NF, Certificate Name, Expiry Date. Expires column uses color coding to indicate near expiry status. all:occm_cert_expiry{namespace="\$namespace"} * 1000 != 0 Expires column:((occm_cert_expiry{namespace="\$namespace"} != 0)-time())*1000

OCCM KPI Dashboard

Figure 8-1 Certificate Expiry Time



Color coding description:-

Red (Critical):- Certificate expiring within 0 <= 7 days Or Certificate expired <= 0 days

Light Red(Major):- Certificate expiring within > 7 <= 30 days

Orange (Minor):- Certificate expiring within > 30 <= 90

Yellow :- Certificate expiring within > 90 <= 180

Green :- Certificates not Expiring sooner

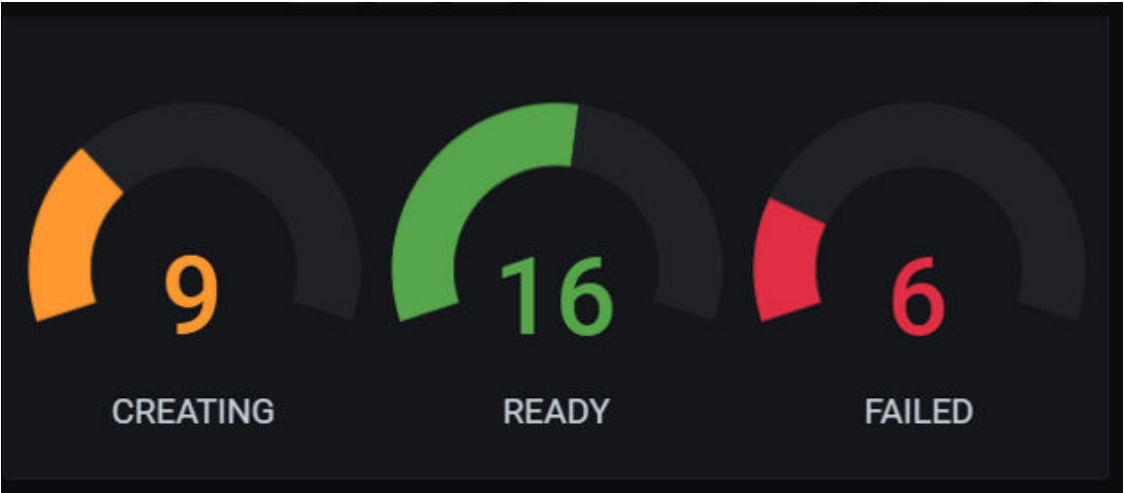
8.2 Certificate Readiness Status

Table 8-2 Certificate Readiness Status

Field	Details
Description	Certificate Readiness Status to indicate if number of Ready and Failed Certificates
Expression	OCCM dashboard in grafana will show Certificate Readiness Status panel gauge visualization to indicate if number of Ready and Failed Certificates Creating:count(occm_cert_status{namespace="\$namespace"} == 1) (Color:Orange) Ready:count(occm_cert_status{namespace="\$namespace"} == 2) (Color:Green) Failed:count(occm_cert_status{namespace="\$namespace"} == 3) (Color:Red) Waiting:count(occm_cert_status{namespace="\$namespace"} == 8) (Color:Light Orange) Expired:count(occm_cert_status{namespace="\$namespace"} == 7) (Color:Red)

OCCM KPI Dashboard

Figure 8-2 Certificate Readiness Status



- Creating: Orange
- Ready: Green
- Failed: Red
- Waiting: Light Orange
- Expired : Red

8.3 CMP Request

Table 8-3 CMP Request

Field	Details
Description	Total CMP requests initiated from OCCM towards CA per NF
Expression	OCCM dashboard in grafana will show CMP Request panel which is total CMP requests per NF. all:sum(rate(occm_cmp_requests_total{namespace="\$namespace"}[2m])) SCP:sum(rate(occm_cmp_requests_total{namespace="\$namespace", nfType=~"SCP scp"}[2m])) NRF:sum(rate(occm_cmp_requests_total{namespace="\$namespace", nfType=~"NRF nrf"}[2m]))

8.4 CMP Responses

Table 8-4 CMP Responses

Field	Details
Description	Total CMP responses received from CA per NF by OCCM
Expression	OCCM dashboard in grafana will show CMP Response panel which is total CMP responses per NF. all:sum(rate(occm_cmp_responses_total{namespace="\$namespace"}[2m])) SCP:sum(rate(occm_cmp_responses_total{namespace="\$namespace", nfType=~"SCP scp"}[2m])) NRF:sum(rate(occm_cmp_responses_total{namespace="\$namespace", nfType=~"NRF nrf"}[2m]))

8.5 Configuration Requests

Table 8-5 Configuration Requests

Field	Details
Description	Total Issuer and Certificate configuration requests
Expression	OCCM dashboard in grafana will show Config Requests panel. Total Issuer and Certificate configuration requests. all:sum(rate(occm_config_http_requests_total{namespace="\$namespace"}[2m])) SCP certs:sum(rate(occm_config_http_requests_total{namespace="\$namespace", uri=~".*/certs.*", nfType=~"SCP scp"}[2m])) NRF certs:sum(rate(occm_config_http_requests_total{namespace="\$namespace", uri=~".*/certs.*", nfType=~"NRF nrf"}[2m])) issuers:sum(rate(occm_config_http_requests_total{namespace="\$namespace", uri=~".*/issuers.*"}[2m]))

8.6 Configuration Responses

Table 8-6 Configuration Responses

Field	Details
Description	Total Issuer and Certificate configuration responses
Expression	OCCM dashboard in grafana will show Config Responses panel. Total Issuer and Certificate configuration responses. all:sum(rate(occm_config_http_responses_total{namespace="\$namespace"}[2m])) SCP certs:sum(rate(occm_config_http_responses_total{namespace="\$namespace", uri=~".*/certs.*", nfType=~"SCP scp"}[2m])) NRF certs:sum(rate(occm_config_http_responses_total{namespace="\$namespace", uri=~".*/certs.*", nfType=~"NRF nrf"}[2m])) issuers:sum(rate(occm_config_http_responses_total{namespace="\$namespace", uri=~".*/issuers.*"}[2m]))

8.7 CPU Usage

Table 8-7 CPU Usage

Field	Details
Description	CPU usage of OCCM pod
Expression	Time series indicates CPU usage of OCCM pod. sum(rate(container_cpu_usage_seconds_total{image!="", namespace="\$namespace", pod=~"occm-.*"}[2m])) by(pod)

8.8 Memory Usage

Table 8-8 Memory Usage

Field	Details
Description	Memory usage of OCCM pod
Expression	Time series indicates Memory usage of OCCM pod. (avg_over_time(container_memory_usage_bytes{container=~"occm", namespace="\$namespace"}[2m]))

8.9 OpenSSL CLI Duration (occm_cmp_cli_durations)

Table 8-9 OpenSSL CLI Duration (occm_cmp_cli_durations)

Field	Details
Description	CMP cli time taken in between request and response from CA
Expression	Used to show the duration of openssl cmp calls occm_cmp_cli_durations_bucket{namespace="occm-ns", uuid="fdsfds-9880-fsd99"}

8.10 Number of requests sent to the CA

Table 8-10 Number of requests sent to the CA

Field	Details
Description	Metric will peg when request cmd prepared and send to CA for generate certificate.
Expression	count(occm_cmp_requests_total{namespace="\$namespace"})

8.11 Number of responses received from CA

Table 8-11 Number of responses received from CA

Field	Details
Description	Metric will peg when response received from CA for generate certificate.
Expression	count(occm_cmp_responses_total{namespace="occm-ns"})

8.12 Number of responses based on response code from CA

Table 8-12 Number of responses based on response code from CA

Field	Details
Description	Metric will peg when response received from CA for generate certificate.
Expression	count(occm_cmp_responses_total{namespace="occm-ns", statusCode="OK", status = "SUCCESS"}) or count(occm_cmp_responses_total{namespace="occm-ns", statusCode="ERR_CMP_COMMAND_FAILED", status = "FAILED"})

8.13 Type of request sent to CA

Table 8-13 Type of request sent to CA

Field	Details
Description	Metric will peg when request cmd prepared and send to CA for generate certificate.
Expression	count(occm_cmp_requests_total{namespace="occm-ns", requestType="ir"}) or count(occm_cmp_requests_total{namespace="occm-ns", requestType="kur"})

8.14 Number of certificates issued by CA

Table 8-14 Number of certificates issued by CA

Field	Details
Description	Metric will peg when response received from CA for generate certificate.

Table 8-14 (Cont.) Number of certificates issued by CA

Field	Details
Expression	count(occm_cmp_responses_total{namespace="occm-ns", status = "SUCCESS", statusCode = "OK"})

8.15 Number of CSRs denied by CA or TLS handshake failures or HTTPs connection failures during CA connection

Table 8-15 Number of CSRs denied by CA or TLS handshake failures or HTTPs connection failures during CA connection

Field	Details
Description	Metric will peg when response received from CA for generate certificate.
Expression	count(occm_cmp_responses_total{namespace="occm-ns", status = "FAILED"}) or count(occm_cmp_responses_total{namespace="occm-ns", statusCode="ERR_CMP_COMMAND_FAILED", status="FAILED"})

8.16 Error while writing the key, certificate, or chain in the Kubernetes secrets

Table 8-16 Error while writing the key, certificate, or chain in the Kubernetes secrets

Field	Details
Description	Metric will peg when cert renew or create worker complete its process
Expression	occm_cert_request_status{namespace="occm-ns", errorReason="ERR_SECRET_FAILED"}

8.17 Unable to access or read from Kubernetes secrets

Table 8-17 Unable to access or read from Kubernetes secrets

Field	Details
Description	Metric will peg when cert renew or create worker complete its process
Expression	occm_cert_request_status_total{namespace="occm-ns", errorReason="ERR_SECRET_EXIST"}

8.18 Check Renewed Certificate

Table 8-18 Check Renewed Certificate

Field	Details
Description	Metric will peg when cert renew or create worker complete its process
Expression	occm_cert_request_status_total{namespace="occm-ns", operationType="RENEW"}

8.19 Certificate Error and Warnings

Table 8-19 Certificate Error and Warnings

Field	Details
Description	List of Certificates having Error and Warnings for duration of 5 mins
Expression	rate(occm_cert_request_status_total{namespace="occm-ns", errorReason!="OK"}[5m])

OCCM KPI Dashboard

Figure 8-3 Certificate Error and Warnings

Certificate Error					
Cert Name	UUID	Operation	Reason	Issuer	nType
OCCM-TLS-Wrong	60761a60-5fe8-4579-a06c-02c549b...	CREATE	ERR_CMP_COMMAND_FAILED	CA-TLS-Wrong	OCCM

Displayed Columns

1. Cert Name - Certificate Name
2. UUID - Certificate UUID
3. Operation - Certificate Operation Type (CREATE or RENEW)
4. Reason - Error code indicating Certificate Error or Warning Reason
5. Issuer - Issuer Name linked to the Certificate

A.1 Creating NF Certificate Using OCCM - Sample Configuration

This section describes the sequence of steps to be performed to generate a signed certificate (NF certificate) using OCCM

1. Create the Issuer:

The following screenshots provide a sample configuration for creating the issuer using CNC Console GUI

a. Figure 4 Create Issuer

UUID:	UUID 0f9d193-fb6-4b1d-a410-a797a8dacc9
Name:	Name CA1
Server URL:	Server URL http://ca1-openssl-mockns1.occm.thruot5.0000
Recipient Distinguished Name:	Recipient Distinguished Name /CN=occm-company.com
Issuer Distinguished Name:	Issuer Distinguished Name /CN=occm-company.com
Total Timeout (Seconds):	Total Timeout (Seconds) 720
Message Timeout (Seconds):	Message Timeout (Seconds) 120

b. Figure 5 CMP Client Authentication Options for OCCM Certificate

CMP Client Authentication Options For OCCM Certificate	
Type:	Type
Digest Algorithm:	Digest Algorithm
MAC Algorithm:	MAC Algorithm
MAC Authentication Input	
Namespace:	Namespace
Secret Name:	Secret Name
Password Key:	Password Key
Reference Key:	Reference Key
Signature Authentication Input	
Namespace:	Namespace
Secret Name:	Secret Name
Key:	Key
Certs:	Cert
Extra Certs:	Extra Certs

c. Figure 6 CMP Client Authentication Options for Other Certificate

CMP Client Authentication Options For Other certificate	
Type:	Type SIGNATURE
Digest Algorithm:	Digest Algorithm SHA256
Signature Authentication Input	
Namespace:	Namespace ns1
Secret Name:	Secret Name ca1-cmp-identity-secret
Key:	Key key-cmpkey.pem
Certs:	Cert ca1-cmpcert.pem
Extra Certs:	Extra Certs
OCCM Trust-Store Secret Input	
Namespace:	Namespace ns1
Name:	Name ca1-occm-trust-store-secret
Root CA Cert:	Root CA Cert rootCA.csr (ca-root.pem x)
Intermediate CA Cert:	Intermediate CA Cert intermediateCA.csr (int-ca-cert.pem x)
Server Cert:	Server Cert servercert.pem

- d. To enable HTTPS communication, provide HTTPS scheme in the server URL field and provide the TLS trust store certificates under TLS config.

Figure 7 HTTPS Scheme

Server URL:

Server URL
https://ca1-openssl.mock.svc.thrust5:8443

Figure 8 Enable TLS Config

▼ TLS Config

Enable TLS: ☒

▼ TLS Trust-Store Secret Input

Namespace:

Namespace
ns1

Name:

Name
ca1-tls-trust-store-secret

TLS Trusted Certs:

TLS Trusted Certs
ManagementCA.pem X

2. **Create OCCM Certificate:**
The following screenshots provide a sample configuration for creating OCCM Certificate using CNC Console GUI. Here, OCCM certificate is configured manually.

a. Figure 9 Create OCCM Certificate

UUID:

UUID
a9410245-449c-4d9a-a60c-9a5796c121cf

Name:

Name
OCCM-CA1

Cert Type:

Cert Type
OCCM

Network Function:

Network Function
OCCM

Purpose:

Purpose
CMP Client Authentication

Issuer:

Issuer
CA1

Life Cycle Management:

Life Cycle Management
MANUAL

Renew Before Expiration (Days):

Renew Before Expiration (Days)
14

b. Figure 10 Private Key Options

▼ Private Key Options

Key Algorithm:

Key Algorithm
RSA

Key Encoding:

Key Encoding
PEM

Key Size:

Key Size
KEYSIZE_2048

▼ Private Key Output

Namespace:

Namespace

Secret Name:

Secret Name

Key:

Key

c. **Figure 11 Public Key Certificate Options**

Public Key Certificate Options

Key Usage

Critical☒

Value(s)

Value(s)

DIGITAL_SIGNATURE X

Extended Key Usage

Critical☐

Value(s)

Value(s)

CLIENT_AUTH X SERVER_AUTH X

Basic Constraints

Critical☐

Value(s)

Value(s)

END_ENTITY

d. **Figure 12 Subject and Subject Alternate Name**

Subject

Country

Country

IN

State

State

KA

Location

Location

BLR

Organization

Organization

Oracle

Organization Unit

Organization Unit

COBU

Common Name

Common Name

o.com.com

Requested Validity (Days)

Requested Validity (Days)

365

Subject Alternate Names

Critical☒

IP Address

IP Address

10.10.10.14 X

DNS Name

DNS Name

x.comany.com X

URI ID API Root

URI ID API Root

URI ID URNs

URI ID URNs

e. **Figure 13 Certificate Output**

Certificate Output

Namespace

Namespace

Secret Name

Secret Name

Key

Key

Certificate Chain Output

Namespace

Namespace

Secret Name

Secret Name

Key

Key

3. **Create NF Certificate:**

The following screenshots provide a sample configuration for creating NF Certificate using CNC Console GUI.

a. **Figure 14 Create NF Certificate**

UUID: b9b1245b-7354-4ba4-a084-f0d5b5ad59fa

Name

Name

NF-TLS-Cert

Cert Type

Cert Type

OTHER

Network Function

Network Function

NF

Purpose

Purpose

NF SBI

Issuer

Issuer

CA1

Life Cycle Management

Life Cycle Management

AUTOMATIC

Override Secret

Override Secret

☒

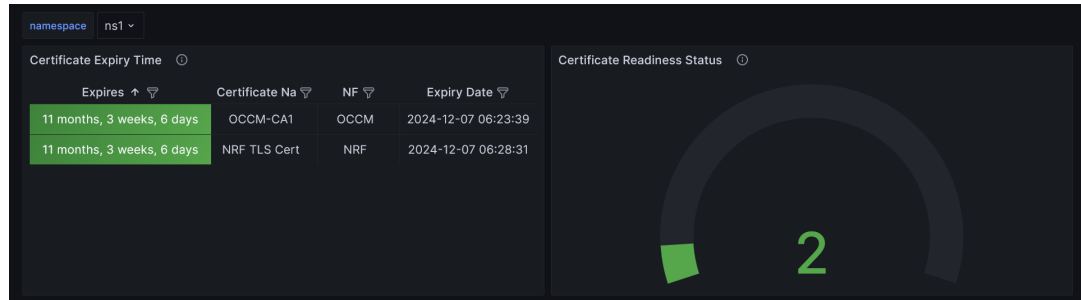
Renew Before Expiration (Days)

Renew Before Expiration (Days)

14

Cloud Native Core, Certificate Management User Guide
F90677-01
Copyright © 2023, 2024, Oracle and/or its affiliates.

September 1, 2025
Page 3 of 6

Figure 19 Sample Grafana Dashboard

The screenshot shows that NRF TLS Cert and CA1 certificates are created successfully. The left panel indicates their expiry time and the right panel shows that both are ready to be consumed.

5. Verify Kubernetes secret

After the certificate request is submitted, verify whether the k8s secret specified under private key output and certificate output location is created or not.

Run the following command to get the content of the Kubernetes secret:

```
kubectl get secret <k8s-secret-name> -n <namespace> -o yaml
```

For example:

```
[scp-user@thrust5-bastion-1 ~]$ kubectl get secret nrf-tls-secret -n ns1 -o yaml
apiVersion: v1
data:
  nrfcert.pem: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCkXXXXXXXXXX
  nrfcertchain.pem: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tXXXXXXXXXX
  nrfkey.pem: LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1XXXXXXXXXXXX
kind: Secret
metadata:
  creationTimestamp: "2023-12-08T06:29:46Z"
  name: nrf-tls-secret
  namespace: ns1
  resourceVersion: "563348905"
  uid: f0eb452d-e977-4809-99b0-c541b154dabe
type: Opaque
```

Output of openssl x509 command for the certificate:

```
kubectl get secret <k8s-secret-name> -n <namespace> -o=go-template='{{index .data "<certificate-output-K8s-secret-key>"}}' | base64 -d | openssl x509 -text -noout
```

For example:

```
[scp-user@thrust5-bastion-1 ~]$ kubectl get secret nrf-tls-secret -n ns1 -o=go-template='{{index .data "nrfcert.pem"}}' | base64 -d | openssl x509 -text -noout
```

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    XXXXXXXXX
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: CN = x.company.com
  Validity
    Not Before: Sep 25 05:46:31 2023 GMT
    Not After : Sep 24 05:46:31 2024 GMT
  Subject: C = IN, ST = KA, L = BLR, O = Oracle, OU = CGBU, CN =
a.company.com
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    Modulus:
      00:c9:1b:35:bf:21:e6:1f:69:9e:78:25:07:4b:6e:
      XXXXXXXXX

      Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Key Usage:
      Digital Signature
    X509v3 Extended Key Usage:
      TLS Web Client Authentication, TLS Web Server Authentication
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Subject Alternative Name:
      IP Address:10.10.10.20, IP Address:10.10.10.21,
DNS:y.commpany.com, DNS:z.commpany.com, URI:urn:uuid:f81d4fae-7dec-11d0-
a765-00a0c91e6bf6
    X509v3 Subject Key Identifier:
      2B:0D:XXXXXXXXXXXX
    X509v3 Authority Key Identifier:
      20:03:XXXXXXXXXXXX
  Signature Algorithm: sha256WithRSAEncryption
  Signature Value:
    XXXXXXXXXXXXXXXXXXXXXXXX
```