Oracle® Communications Offline Mediation Controller

Cloud Native Installation and Administration Guide

Release 15.0
F87496-02
June 2024

ORACLE

Oracle Communications Offline Mediation Controller Cloud Native Installation and Administration Guide, Release 15.0

F87496-02

Contents

Preface

This guide provides general information on how to configure and install Oracle Communications Offline Mediation Controller in a cloud native environment.

Audience

This document is intended for DevOps administrators and those involved in installing and maintaining an Oracle Communications Offline Mediation Controller cloud native deployment.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Overview of the Offline Mediation Controller Cloud Native Deployment

Oracle Communications Offline Mediation Controller can be configured to run as a cloud native application in a containerized and orchestrated deployment architecture.

Topics in this document:

About the Offline Mediation Controller Cloud Native Deployment

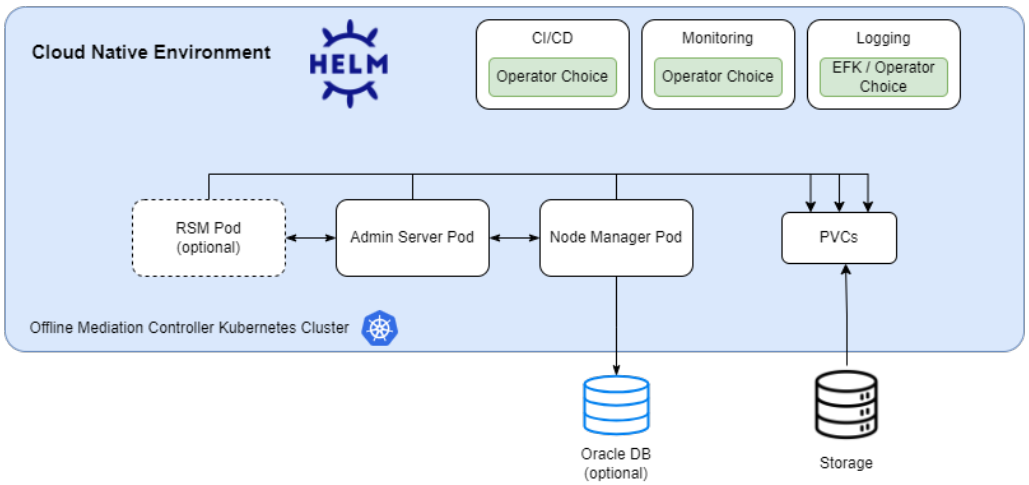
Oracle Communications Offline Mediation Controller supports its deployment on a cloud native environment. This allows you to harness the benefits of cloud with the services of Offline Mediation Controller. For more information about Offline Mediation Controller, see "Overview of Offline Mediation Controller" in *Offline Mediation Controller User's Guide*.

You can also set up your own cloud native environments. You use the cloud native deployment package to automate the deployment of Offline Mediation Controller products and speed up the process to get services up and running, with product deployments preconfigured to communicate with each other through Helm charts.

Offline Mediation Controller Cloud Native Deployment Architecture

Figure 1-1 shows the pods and other components in a typical Offline Mediation Controller cloud native deployment.

Figure 1-1 Offline Mediation Controller Cloud Native Deployment Architecture

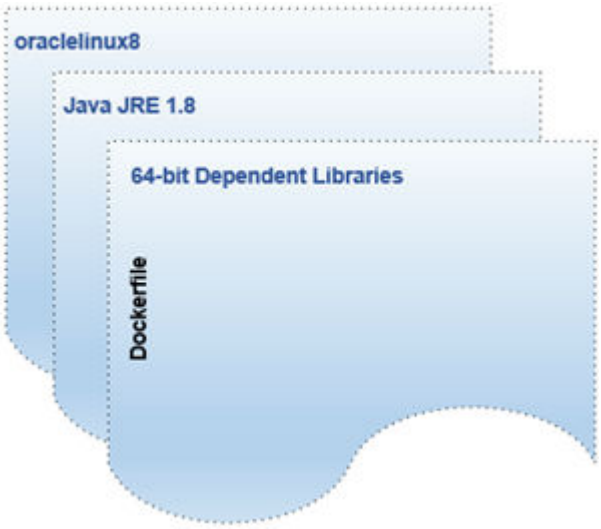


Images and Containers

The Offline Mediation Controller cloud native deployment image has been designed and hardened to serve only the purpose it's supposed to. The image is built by stacking multiple layers, extending an operating system image with a dependent library image, and then with an image packing the application.

The Offline Mediation Controller image uses the layering pattern shown in Figure 1-2. If you want to build your own Offline Mediation Controller image, you must layer the images in this pattern.

Figure 1-2 Base Image Layering



2
Planning Your Installation

The Oracle Communications Offline Mediation Controller cloud native deployment package includes Docker images and Helm charts to help you deploy and manage pods of product services in Kubernetes.

Topics in this document:

Overview of the Offline Mediation Controller Deployment Package

The Offline Mediation Controller cloud native deployment package includes the following:

- Ready-to-use images and Helm charts to help you orchestrate containers in Kubernetes.
- Sample Dockerfiles and scripts that you can use as a reference for building your own images.

You can use the images and Helm charts to help you deploy and manage pods of Offline Mediation Controller product services in Kubernetes. Communication between pods of services of Offline Mediation Controller products is preconfigured in the Helm charts.

About Offline Mediation Controller Pods and Images

Table 2-1 lists the pods and images for Offline Mediation Controller whose containers are created and services are exposed through them. For the image name, replace 15.0.0.*x*.0 with the patch set version number, such as 15.0.0.1.0.

Table 2-1 Offline Mediation Controller Pods and Images

Pod	Replica Type	Image Name	Container Port
admin-server-app-58dbd45d58-9vqll	Single	**oc-cn-ocomc:15.0.0.*x*.0**	55105
node-mgr-app-5577bbb854-76rx4	Single	**oc-cn-ocomc:15.0.0.*x*.0**	55109
ocomc-rsm-5c7848989c-fhdkc	Multiple	**oc-cn-ocomc-rsm:15.0.0.*x*.0**	8080

About Offline Mediation Controller Services

Table 2-2 lists the services for Offline Mediation Controller.

Table 2-2 Offline Mediation Controller Services

Service	Service Type	Port	Notes
admin-server-app	ClusterIP	55105	Client on same worker node
node-mgr-app	ClusterIP	55109	Client on same worker node
admin-server-app-external	NodePort	External port	Client on Windows system
node-mgr-app-external	NodePort	External port	Client on Windows system
ocomc-rsm	NodePort	8080	Client on Windows System

To connect to the Administration Server running in a Kubernetes cluster, you must install the Administration Client outside of the Kubernetes cluster and then connect it to the service and port of the Administration Server.

- If the Administration Client is installed on the same node where the Administration Server pod is running, use a clusterIP service type with the Administration Server.
- If the Administration Client is located remotely or is on a Windows system, use a NodePort service type with the Administration Server.

For more information about connecting the Administration Client, see "Connecting Your Administration Client".

3

Preparing Your Offline Mediation Controller Cloud Native Environment

You prepare your system for the Oracle Communications Offline Mediation Controller cloud native deployment by installing all prerequisite software and downloading the Offline Mediation Controller Helm charts and images.

Topics in this document:

Tasks for Preparing Your Offline Mediation Controller Cloud Native Environment

To prepare your system for the Offline Mediation Controller cloud native deployment:

1. If you want to integrate Offline Mediation Controller with Elastic Charging Engine (ECE) and the Billing and Revenue Management (BRM) cloud native deployment package, set up the ECE and BRM cloud native deployment prior to setting up Offline Mediation Controller. See *BRM Cloud Native Deployment Guide*.
2. Set up your Offline Mediation Controller environment by installing and configuring all prerequisite software. See "Setting Up Your Environment".
3. Download the Helm charts for the Offline Mediation Controller cloud native deployment. See "Downloading Packages for the Offline Mediation Controller Cloud Native Helm Charts".
4. Download the Offline Mediation Controller cloud native images in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling Offline Mediation Controller Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading Offline Mediation Controller Images from Oracle Software Delivery Website".

Setting Up Your Environment

Set up your environment with the following technologies installed, configured, and tuned for performance, networking, security, and high availability. Make sure backup nodes are available in case of system failure in any of the cluster's active nodes.

- **Podman:** The Podman tool is used to containerize Offline Mediation Controller products.

For more information, see the Podman documentation (https://docs.podman.io/en/latest/index.html).

- **Kubernetes:** Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. It groups containers into logical units for easy management and discovery. When you deploy Kubernetes, you get a physical cluster with machines called nodes. A reliable cluster must have multiple worker nodes spread over separate physical infrastructures, and a very reliable cluster must have multiple primary nodes spread over different physical infrastructures.

  Set up a Kubernetes cluster for your BRM cloud native deployment, securing access to the cluster and its objects with the help of service accounts and proper authentication and authorization modules. Also, set up the following in your cluster:

  – **Volumes:** A container's file system lives only as long as the container does. When a container terminates and restarts, file system changes are also lost. You shouldn't access the container file system or pods frequently, and sharing data between container and host systems is not easy. Volumes appear as a directory in the container file system and provide a way to share data. The Offline Mediation Controller cloud native deployment package uses persistent volumes for sharing data in and out of containers but doesn't enforce any particular type. You can choose from the volume type options available in Kubernetes.

    You can choose an external incubator to create persistent volumes, but ensure it supports the ReadWriteMany access mode and PVC sharing between pods.

  – **A networking model:** Kubernetes assumes that pods can communicate with other pods, regardless of which host they land on. Every pod has a different IP address, so you don't need to explicitly create a link between pods. You rarely need to deal with mapping container ports to host ports. While Kubernetes doesn't offer a solution to support its assumption, several implementations meet the fundamental requirements of Kubernetes' networking model. Choose the networking element depending on the cluster requirement.

  For more information about Kubernetes, see *Kubernetes Concepts* (https://kubernetes.io/docs/concepts/).

  > **Note:**
  >
  > Secure your cluster according to standard DevOps practices.

- **Fluentd:** Fluentd forms your logging layer, collecting log files from your Offline Mediation Controller service pods and transforming them. The Fluentd-concat plugin is used to concatenate multiline log files. You set up Fluentd on your Kubernetes nodes. Configure all applications to redirect their logs to STDOUT so Fluentd can parse your log files.

  For more information about Fluentd, see *Fluentd Overview* (https://docs.fluentd.org/quickstart).

- **Helm:** Helm is a package manager that helps you install and maintain software on a Kubernetes system. In Helm, a package is called a *chart*, which consists of YAML files and templates rendered into Kubernetes manifest files. The BRM cloud native deployment package includes Helm charts that help create Kubernetes objects, such as ConfigMaps, Secrets, controller sets, and pods, with a single command.

  For more information about Helm, see *Helm Introduction to Helm* (https://helm.sh/docs/using_helm/).

- **Oracle Database:** An Oracle database must be installed and accessible through the Kubernetes network so that the pods can perform database operations. It can be either a CDB or a non-CDB.

For the complete list of software compatible with the Offline Mediation Controller cloud native deployment package, see "Offline Mediation Controller Cloud Native Deployment Software Compatibility" in *Offline Mediation Controller Compatibility Matrix*.

Downloading Packages for the Offline Mediation Controller Cloud Native Helm Charts

To download the Offline Mediation Controller cloud native Helm charts:

1. Go to https://edelivery.oracle.com.
2. Sign in to the Oracle Software Delivery website using an Oracle account.
3. Search for and select **Oracle Communications Offline Mediation Controller Cloud Native Deployment Option 15.0.0.*x*.0** and then click **Continue**.
4. Make sure all packages are selected, and then click **Continue**.
5. Accept the Oracle standard terms and restrictions, and then click **Continue**.
6. Select the **Oracle Communications Offline Mediation Controller Cloud Native Deployment Option Helm Chart 15.0.0.*x*.0** package and then click **Download**.

   The package is downloaded to a Zip file.
7. Extract the Offline Mediation Controller Helm chart archive file (**oc-cn-ocomc-15.0.0.*x*.0.tgz**) from the Zip file.
8. Extract the Helm chart from the archive file by running this command:

   ```
   tar xvzf oc-cn-ocomc-15.0.0.x.0.tgz
   ```

   The **oc-cn-ocomc** directory is extracted from the archive file.

Pulling Offline Mediation Controller Images from the Oracle Container Registry

To pull Offline Mediation Controller cloud native images from the Oracle Container Registry, do the following:

1. In a web browser, go to https://container-registry.oracle.com.
2. Sign in to the Oracle Container Registry using an Oracle account.

   > **Note:**
   >
   > To pull images for licensed software on the Oracle Container Registry, you must have an Oracle account. You can create an Oracle account at https://profile.oracle.com/myprofile/account/create-account.jspx.

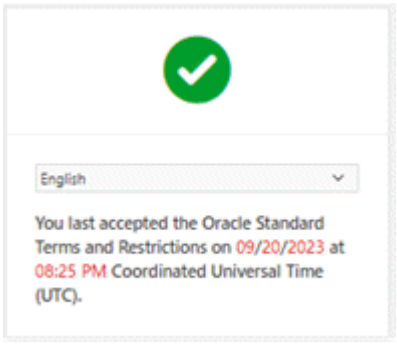
3. Select the **Oracle Communications Cloud Scale Monetization** container.

   The Oracle Communications Cloud Scale Monetization page appears.

4. Click one of the following repository names:
   - **oc-cn-ocomc**: Offline Mediation Controller image
   - **oc-cn-ocomc-rsm**: Offline Mediation Controller REST Services Manager image

   The repository page appears.
5. Accept the Oracle terms and restrictions by:
   a. For non-CPU repositories, selecting your desired language.
   b. Clicking **Continue**.
   c. Scrolling to the bottom of the terms and restrictions page, and clicking **Accept**.

   If successful, you will see something similar to this:



6. On your host system, log in to the Oracle Container Registry using the Podman command-line interface (CLI):

   `podman login container-registry.oracle.com`
7. When prompted for a user name and password, enter your Oracle credentials.
8. Pull the Offline Mediation Controller cloud native image from the registry:

   `podman pull container-registry.oracle.com/communications_monetization/`*`imageName`*`:`*`tag`*

   where:
   - *imageName* is the name of the software image: **oc-cn-ocomc** or **oc-cn-ocomc-rsm**.
   - *tag* is the tag name for the image, such as **15.0.0.***x***.0**.

   For example, to pull the Offline Mediation Controller cloud native image from the registry:

   ```
   podman pull container-registry.oracle.com/communications_monetization/oc-cn-
   ocomc:15.0.0.x.0
   ```

   The image is pulled from the Oracle Container Registry and stored locally, where it is ready to be used to deploy containers.
9. Confirm the images were pulled from the Oracle Container Registry:

   `podman images`

   If successful, you will see something similar to this:

   ```
   REPOSITORY
   TAG            IMAGE ID        CREATED
   container-registry.oracle.com/communications_monetization/oc-cn-ocomc
   15.0.0.x.0     133dd3580b87    2 seconds ago
   ```

```
container-registry.oracle.com/communications_monetization/oc-cn-ocomc-rsm
15.0.0.x.0     136dd3593b47    3 seconds ago
```

10. Log out of the registry to prevent unauthorized access and to remove any record of sign-in credentials that Podman might store for future operations:

    podman logout container-registry.oracle.com

Downloading Offline Mediation Controller Images from Oracle Software Delivery Website

To download Offline Mediation Controller cloud native images from the Oracle Software Delivery website:

1. Go to https://edelivery.oracle.com.
2. Sign in to the Oracle Software Delivery website using an Oracle account.
3. Search for and select **Oracle Communications Offline Mediation Controller Cloud Native Deployment Option 15.0.0.*x*.0**.
4. Download Zip files for the following:
   - Oracle Communications Offline Mediation Controller Cloud Native Deployment Option REST Services Manager 15.0.0.*x*.0
   - Oracle Communications Offline Mediation Controller Cloud Native Deployment Option 15.0.0.*x*.0
5. From the Zip files, extract the following archive files:
   - Offline Mediation Controller image (**oc-cn-ocomc-15.0.0.*x*.0.tar**)
   - Offline Mediation Controller REST Services Manager image (**oc-cn-ocomc-rsm-15.0.0.*x*.0.tar**)
6. Load the files as images into your Podman system using the following command:

   podman load --input *fileName*

   where *fileName* is **oc-cn-ocomc-15.0.0.*x*.0.tar** or **oc-cn-ocomc-rsm-15.0.0.*x*.0.tar**.

If you use an internal registry to access images from different Kubernetes nodes, push the images from your local system to the registry server. For example, if the registry is identified by *RepoHost*:*RepoPort*, you would push the Offline Mediation Controller REST Services Manager image to the registry using the Podman CLI like this:

1. Tag the Offline Mediation Controller REST Services Manager image with the registry server:

   podman tag ocomc-rsm:15.0.0.*x*.0 *RepoHost*:*RepoPort*/**oc-cn-ocomc-rsm:15.0.0.*x*.0**

2. Push the image to the registry server:

   podman push *RepoHost*:*RepoPort*/**oc-cn-ocomc-rsm:15.0.0.*x*.0**

4

Installing the Offline Mediation Controller Cloud Native Deployment Package

Learn how to install the Oracle Communications Offline Mediation Controller cloud native deployment package on a cloud native environment.

Topics in this document:

About Deploying into Kubernetes

Helm is the recommended package manager for deploying Offline Mediation Controller cloud native services into Kubernetes. A Helm chart is a collection of files that describe a set of Kubernetes resources. It includes YAML template descriptors for all Kubernetes resources and a **values.yaml** file that provides default configuration values for the chart.

The Offline Mediation Controller cloud native deployment package includes **oc-cn-ocomc-helm-chart-15.0.0.*x*.0.tgz**.

When you install the Helm chart, it generates valid Kubernetes manifest files by replacing default values from **values.yaml** with custom values from **override-values.yaml**, and creates Kubernetes resources. Helm calls this a new release. You use the release name to track and maintain this installation.

Automatically Pulling Images from Private Docker Registries

You can automatically pull images from your private Docker registry by creating an **ImagePullSecrets**, which contains a list of authorization tokens (or Secrets) for accessing a private Docker registry. You then add references to the **ImagePullSecrets** in your Offline Mediation Controller Helm chart's **override-values.yaml** file. This allows pods to submit the Secret to the private Docker registry whenever they want to pull images.

Automatically pulling images from a private Docker registry involves these high-level steps:

1. Create a Secret outside of the Helm chart by entering this command:

   ```
   kubectl create secret docker-registry SecretName --docker-
   server=RegistryServer --docker-username=UserName --docker-
   password=Password -n NameSpace
   ```

   where:

- *SecretName* is the name of your Kubernetes Secret.
- *RegistryServer* is your private Docker registry's fully qualified domain name (FQDN) (*repoHost*:*repoPort*).
- *UserName* and *Password* are your private Docker registry's user name and password.
- *NameSpace* is the namespace you will use for installing the Offline Mediation Controller Helm chart.

For example:

```
kubectl create secret docker-registry my-docker-registry --docker-
server=example.com:2660/ --docker-username=xyz --docker-password=password -n oms
```

2. Add the **imagePullSecrets** key to your **override-values.yaml** file for **oc-cn-ocomc**:

   imagePullSecrets: *SecretName*

3. Add the **ocomc.imageRepository** key to your **override-values.yaml** file:

   imageRepository: "*RegistryServer*"

4. Deploy **oc-cn-ocomc**.

Automatically Rolling Deployments by Using Annotations

Whenever a ConfigMap entry or a Secret file is modified, you must restart its associated pod. This updates the container's configuration, but the application is notified about the configuration updates only if the pod's deployment specification has changed. Thus, a container could be using the new configuration, while the application keeps running with its old configuration.

You can configure a pod to automatically notify an application when a Container's configuration has changed. To do so, configure a pod to automatically update its deployment specification whenever a ConfigMap or Secret file changes by using the **sha256sum** function. Add an **annotations** section similar to this one to the pod's deployment specification:

```
kind: Deployment
spec:
  template:
    metadata:
      annotations:
        checksum/config: {{ include (print $.Template.BasePath "/
configmap.yaml") . | sha256sum }}
```

For more information, see *Chart Development Tips and Tricks* in the Helm documentation (https://helm.sh/docs/howto/charts_tips_and_tricks/#automatically-roll-deployments).

About the Offline Mediation Controller Pods

Table 4-1 lists the PersistentVolumeClaims (PVCs) used by the Offline Mediation Controller server.

Table 4-1 List of PVCs in Offline Mediation Controller Server

PVC Name	Default Pod Internal File System
admin-server-pvc	*OMC_home*/**config/adminserver**

Table 4-1 (Cont.) List of PVCs in Offline Mediation Controller Server

PVC Name	Default Pod Internal File System
node-manager-pvc	*OMC_home*
vol-keystore	**/home/ocomcuser/keystore**
vol-data	**/home/ocomcuser/data**
vol-external	**/home/ocomcuser/ext**
vol-suspense	**/home/ocomcuser/suspense**

To share these PVCs between Offline Mediation Controller pods, you must use a persistent volume provisioner that:

- Provides ReadWriteMany access and sharing between the pods.
- Mounts all external volumes with a user (**ocomcuser**) that has a UID and GID of 1000 and that has full permissions.
- Has its volume reclaim policy set to avoid data and configuration loss in a mounted file system.
- Is configured to share data, external KeyStore volumes, and wallets between Offline Mediation Controller pods and the Administration Client.

You must place all CDR files inside the vol-data PVC and then configure the internal file system path of the vol-data PVC in your Administration Client. All CDRs must have read and write permission for the **ocomcuser** user.

You must place all necessary third-party and cartridge JAR files in a **3pp** and **cartridges** directory inside the vol-external PVC, and then restart the pods. After the PVC is mounted, these cartridges will be available in each pod at **/home/ocomcuser/ext/3pp** and **/home/ocomcuser/ext/cartridges**.

The Offline Mediation Controller wallet files will be created and used through the shared vol-keystore PVC.

ECE-related configuration inside the **UDCEnvironment** file for the Administration Client must refer to the internal path of the pod.

Inside deployment templates, nodeSelector can be set in the pod's specification with the worker node having mounted PVC and the Administration Client installed.

Configuring Offline Mediation Controller Services

The Offline Mediation Controller unified Helm chart (**oc-cn-ocomc**) configures and deploys all of your product services. YAML descriptors in the **oc-cn-ocomc/templates** directory use the **oc-cn-ocomc/values.yaml** file for most of the values. You can override the values by creating an **override-values.yaml** file.

The unified Helm chart includes both Offline Mediation Controller Core and REST Services Manager under a single Helm chart. It contains both Core and RSM Helm charts as subcharts within it. You can use the following keys to toggle deployment between Offline Mediation Controller Core or REST Services Manager by setting their values to either **true** or **false**:

- Use **charts.enableCore** to enable Offline Mediation Controller Core.
- Use **charts.enableRSM** to enable Offline Mediation Controller RSM.

Table 4-2 lists the keys that directly impact Offline Mediation Controller services. Add these keys to your **override-values.yaml** file with the same path hierarchy.

> **Note:**
>
> - If you are using a Windows-based client, the **adminsvrIp**, **nmExternalPort**, **adminsvrExternalPort**, and **adminsvrFirewallPort** keys must be set. To connect with the Windows-based client, use external services with a NodePort type. In this case, the **adminsvrIp** will be the worker node IP. Restart the pod after setting **adminsvrIp**.
> - If graphical desktop support such as VNC is available on a worker node, the client can be installed on the same worker node in which Administration Server and Node Manager pods are running. In this case, set the service type to ClusterIP and do not set the **nmExternalPort**, **adminsvrExternalPort**, and **adminsvrFirewallPort** keys.

Table 4-2 Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
fromRelease	**ocomcCore.upgrade**	The latest base version of Offline Mediation Controller that you have installed on your system in the format **15.0.0.*x*.0**. When upgrading from one release to another, this value must be populated. When not performing an upgrade, leave this value empty.
imagePullSecrets	**ocomcCore**	The location of your **imagePullSecrets**, which stores the credentials (or Secret) for accessing your private Docker registry. See "Automatically Pulling Images from Private Docker Registries".

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description

ece.*	**ocomcCore**	The details for connecting to ECE. Add these keys only if you are integrating Offline Mediation Controller with ECE:
		• **imageRepository**: The Docker registry URL for the ECE image. The default is **oc-cn-ece**.
		• **deployment.ecs.imageName**: The name of the ECE image.
		• **deployment.ecs.imageTag**: The tag name for the ECE image.
		• **deployment.ecs.imagePullPolicy**: The pull policy of the ECE image. The default value is **IfNotPresent**, which specifies not to pull the image if it's already present. Applicable values are **IfNotPresent** and **Always**.
		• **deployment.ecs.clusterName**: The ECE cluster name. The default is **BRM**.
		• **deployment.ecs.serviceName**: The ECE service name. The default is **ece-server**.
		• **deployment.ecs.persistenceEnabled**: Whether ECE will persist its cache data in the Oracle database: **true** or **false**. The default is **false**.
		• **deployment.ecs.coherenceClusterPort**: The optional value indicating the Coherence port used by the ECE component.
		For example:
		`ece:`
		`  imageRepository: ""`
		`  deployment:`
		`  ecs:`
		`  imageName: "oc-cn-ece"`
		`  imageTag: "15.0.0.x.0"`
		`  imagePullPolicy: IfNotPresent`
		`  clusterName: BRM`
		`  serviceName: ece-server`
		`  persistenceEnabled: "true"`
		`  coherenceClusterPort: ""`

serviceMonitor.enabled	**ocomcCore.ocomc**	Whether to automatically scrape Offline Mediation Controller metrics and send them to Prometheus Operator. See "Enabling the Automatic Scraping of Metrics".
imageRepository	**ocomcCore.ocomc**	The Docker registry URL for the Offline Mediation Controller image.
name	**ocomcCore.ocomc.secretEnv**	The name of your Offline Mediation Controller Secret, such as **ocomc-secret-env**.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
uniPass	**ocomcCore.ocomc.secretEnv**	Use this key to apply a uniform password to all Offline Mediation Controller cloud native services, including: • Database Schemas • Offline Mediation Controller Root Login • Oracle Wallets To override this password for a specific service, specify a different password in the service's key. **Note**: Use this key for test or demonstration systems only.
nmKeypass	**ocomcCore.ocomc.secretEnv**	The password for the Node Manager domain SSL identity key.
nmKeystorepass	**ocomcCore.ocomc.secretEnv**	The Offline Mediation Controller Secrets required for SSL and installation.
adminKeypass	**ocomcCore.ocomc.secretEnv**	The password for the Administration Server domain SSL identity key.
adminKeystorepass	**ocomcCore.ocomc.secretEnv**	The password for the Administration Server domain SSL identity store.
walletPassword	**ocomcCore.ocomc.secretEnv**	The string password for opening the wallet.
ocomcPassword	**ocomcCore.ocomc.secretEnv**	The Offline Mediation Controller password.
adminServerPassword	**ocomcCore.ocomc.secretEnv**	The Administration Server password.
restart_count	**ocomcCore.ocomc.configEnv**	Increment the existing value by 1 to re-create pods using the **helm upgrade** command.
name	**ocomcCore.ocomc.configEnv**	The name of your Offline Mediation Controller ConfigMap, such as **ocomc-configmap-env**.
sslEnabled	**ocomcCore.ocomc.configEnv**	Whether SSL is enabled in your Offline Mediation Controller cloud native environment: **true** or **false**.
eceEnabled	**ocomcCore.ocomc.configEnv**	Whether ECE is deployed and enabled in your cloud native environment: **true** or **false**.
ecePath	**ocomcCore.ocomc.configEnv**	The directory in which you installed ECE in your Offline Mediation Controller cloud native environment. Set this key to **/home/ocomcuser/install/**, unless you are creating custom images.
walletFolder	**ocomcCore.ocomc.configEnv**	The location of the Oracle wallet, which contains your TLS certificates. For example: **/home/ocomcuser/ext/**.
thirdPartyFolder	**ocomcCore.ocomc.configEnv**	The directory in which you installed the third-party software required by Offline Mediation Controller. For example: **/home/ocomcuser/ext/3pp**. See "Offline Mediation Controller Cloud Native Deployment Software Compatibility" in *Offline Mediation Controller Compatibility Matrix*.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
cartridgeFolder	**ocomcCore.ocomc.configEnv**	The directory in which you installed your Offline Mediation Controller cartridge packs. Set this key to **/home/ocomcuser/ext/cartridges**, unless you are creating custom images.
nmDebug	**ocomcCore.ocomc.configEnv**	The Node Manager debug mode. The default is **false**.
nmPort	**ocomcCore.ocomc.configEnv**	The Node Manager port. The default is **55109**.
nmExternalPort	**ocomcCore.ocomc.configEnv**	The external port for the Node Manager. It must be in the range specified for the Kubernetes NodePort service. Set this key only if Administration Client is installed remotely or on a Windows system. See "Connecting Your Administration Client".
nmIp	**ocomcCore.ocomc.configEnv**	The external IP for the Node Manager. This key is required only if Node Manager needs to run with a specific external IP. The default is **node-mgr-app**.
metricsPortCN	**ocomcCore.ocomc.configEnv**	The port number at which Node-Manager-level metrics are exposed in Prometheus format. The default is **32000**. The port number must be in the range specified for the Kubernetes NodePort service. See "Using Prometheus Operator to Monitor Offline Mediation Controller Cloud Native" for more information.
metricsPort	**ocomcCore.ocomc.configEnv**	The port number at which JMV metrics are exposed in Prometheus format. The default is **8082**. See "Using Prometheus Operator to Monitor Offline Mediation Controller Cloud Native" for more information.
nmKeystorePath	**ocomcCore.ocomc.configEnv**	The path to the Node Manager domain KeyStore files. Set this key to **/home/ocomcuser/keystore/**, unless you are creating custom images.
nmDname	**ocomcCore.ocomc.configEnv**	The distinguished name (DN) for Node Manager. The default is **CN=$HOSTNAME,OU=OracleCloud,O=OracleCorporation,L=RedwoodShores,S=California,C=US**.
nmsslAlias	**ocomcCore.ocomc.configEnv**	The SSL alias for Node Manager. The default is **nodeManager**.
nmKeystoreValidity	**ocomcCore.ocomc.configEnv**	The number of days the SSL KeyStore certificate for Node Manager will be valid, such as **365** for one year.
adminsvrPort	**ocomcCore.ocomc.configEnv**	The port number for the Administration Server.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
adminsvrExternalPort	**ocomcCore.ocomc.configEnv**	The external port for the Administration Server. It must be in the range specified for the Kubernetes NodePort service. Set this key only if Administration Client is installed remotely or on a Windows system. See "Connecting Your Administration Client".
adminsvrFilewallPort	**ocomcCore.ocomc.configEnv**	The Administration Server firewall port. It must be in the range specified for the Kubernetes NodePort service. Set this key only if Administration Client is installed remotely or on a Windows system. See "Connecting Your Administration Client".
adminsvrIp	**ocomcCore.ocomc.configEnv**	The external Administration Server IP (that is, the worker node host IP where the Administration Server pod is scheduled). See "Connecting Your Administration Client".
adminsvrNodeSelectorHostName	**ocomcCore.ocomc.configEnv**	The host name of the Administration Server node selector.
adminsvrNodeSelectorIp	**ocomcCore.ocomc.configEnv**	The external IP address of the Administration Server node selector.
adminsvrNodeSelector	**ocomCore.ocomc.configEnv**	The name of the Administration Server node selector.
adminsvrAuthMode	**ocomcCore.ocomc.configEnv**	Whether the Administration Server requires authorization. The default is **false**.
adminsvrDebug	**ocomcCore.ocomc.configEnv**	Whether the Administration Server debug mode is turned on. The default is **false**.
adminsvrCartridgeFolder	**ocomcCore.ocomc.configEnv**	The name of the directory in which the cartridge pack JAR files reside.
adminsvrPatchFolder	**ocomcCore.ocomc.configEnv**	The name of the directory in which your **UDCEnvironment** file and schema files reside.
adminsvrRuleFolder	**ocomcCore.ocomc.configEnv**	The name of the directory in which your rule file resides.
adminsvrSharedRuleFolder	**ocomcCore.ocomc.configEnv**	The name of the shared rule directory.
adminsvrDshost	**ocomcCore.ocomc.configEnv**	The host name of the Administration Server DS. The default is **localhost**.
adminsvrDsport	**ocomcCore.ocomc.configEnv**	The Administration Server DS port. The default is **13001**.
adminsvrAuthuser	**ocomcCore.ocomc.configEnv**	Whether the Administration Server must authenticate users.
adminsvrLdapurl	**ocomcCore.ocomc.configEnv**	The URL and the LDAP listening port of the Oracle Unified Directory system.
adminsvrLdapdomain	**ocomcCore.ocomc.configEnv**	The base DN for the LDAP server.
adminsvrGrpinfo	**ocomcCore.ocomc.configEnv**	The **AdminServerImpl.properties** parameter used by the Administration Server.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
adminsvrAdmindn	**ocomcCore.ocomc.configEnv**	The DN for the Administration Client. For example: uid=Admin,ou=People.
adminsvrMemberval	**ocomcCore.ocomc.configEnv**	The **AdminServerImpl.properties** parameter used by the Administration Server.
adminsvrTimeout	**ocomcCore.ocomc.configEnv**	The session timeout, in minutes, between the Administration Server and Administration Client. The default is **30**.
adminsvrDisasterRecovery	**ocomcCore.ocomc.configEnv**	Whether to configure the ECE DC pod for disaster recovery. The default is **false**.
adminsvrHostBased	**ocomcCore.ocomc.configEnv**	The ability to scale up the number of Node Manager pods throughout the lifecycle of the pods. See "Scaling Up Node Manager Pods". • **true**: You can scale up the Node Manager pods to meet your current capacity requirements. This is the default. • **false**: You will not be able to scale up the Node Manager pods. **Note**: You cannot change this key's value after a Node Manager has been added to the Administration Server. To change the setting, you must create a fresh installation of Offline Mediation Controller.
forceWaitForNARSToBeProcessed	**ocomcCore.ocomc.configEnv**	When scaling down Node Manager pods, this value must be set to **true**. This specifies to wait until the EP and DC nodes finish processing all network account records (NARs) that are already present in their input before shutting down cartridges. For all other cases, set this key to **false**.
waitForNarsToProcessInSecs	**ocomcCore.ocomc.configEnv**	The amount of time, in seconds, to wait for NARs to reach the input of a cartridge and then be processed. Configure this based on the time the previous cartridge takes to write out its NARs. The default is **10**. The maximum allowed value is 60.
adminsvrTruststorePath	**ocomcCore.ocomc.configEnv**	The path to your Administration Server domain SSL TrustStore file. Set this key to **/home/ocomcuser/ext/**, unless you are creating custom images.
adminsvrKeystorePath	**ocomcCore.ocomc.configEnv**	The path to your Administration Server domain SSL KeyStore file. Set this key to **/home/ocomcuser/keystore/**, unless you are creating custom images.
adminsvrDname	**ocomcCore.ocomc.configEnv**	The DN for the Administration Server. The default is **CN=$HOSTNAME,OU=OracleCloud,O=OracleCorporation,L=RedwoodShores,S=California,C=US**.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
adminsvrsslAlias	**ocomcCore.ocomc.configEnv**	The alias name for the Administration Server.
adminsvrKeystoreValidity	**ocomcCore.ocomc.configEnv**	The number of days the SSL KeyStore certificate for the Administration Server will be valid, such as **365** for one year.
adminclientTruststorePath	**ocomcCore.ocomc.configEnv**	The path to your Administration Client domain SSL TrustStore file. Set this key to **/home/ocomcuser/keystore/**, unless you are creating custom images.
ocomcSoftwarePath	**ocomcCore.ocomc.configEnv**	Set this key to **/container-scripts/OCOMC-15.0.0.*x*.0_generic_full.jar**, unless you are creating custom images.
ocomcSoftwareUpgradePath	**ocomcCore.ocomc.configEnv**	If you are upgrading from a previous release to 15.0, set this to **/container-scripts/OCOMC-15.0.0.*x*.0_generic_full.jar**, where *x* is the patch set version you are upgrading to. Otherwise, leave this key empty.
inventoryFilePath	**ocomcCore.ocomc.configEnv**	The inventory file path.
oudRootUserDn	**ocomcCore.ocomc.configEnv**	The DN for the Oracle Unified Directory root user. The default is **cn=Directory Manager**.
oudPath	**ocomcCore.ocomc.configEnv**	The path to the Oracle Unified Directory.
oudLdapPort	**ocomcCore.ocomc.configEnv**	The port number on which the LDAP server is listening. The default is **1389**.
oudBaseDn	**ocomcCore.ocomc.configEnv**	The DN for the Oracle Unified Directory. The default is **dc=ocomcexample.com**.
adminConnectPort	**ocomcCore.ocomc.configEnv**	The Administration Server port for the Oracle Unified Directory. The default is **4444**.
hostName	**ocomcCore.ocomc.configEnv**	The host name of the server on which Offline Mediation Controller is deployed. The default is **localhost**.
oracleHome	**ocomcCore.ocomc.configEnv**	The path where you want to install Offline Mediation Controller. Set this key to **/home/ocomcuser/install**, unless you are creating custom images.
forceGenSslcert	**ocomcCore.ocomc.configEnv**	Whether to regenerate the SSL certificate when the pod restarts. The default is **false**.
oPatch	**ocomcCore.ocomc.configEnv**	The OPatch number you want to apply. Copy the OPatch Zip file in **/home/ocomcuser/ext** if the **oraInventory** directory is present in the Offline Mediation Controller installation. If not, unzip the patch file in **/home/ocomcuser/ext/** and give permissions to the folder. Populate this field only when an installation is already present and you want to apply OPatch. Otherwise, keep it empty.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
testNodeChain.enabled	**ocomcCore.ocomc**	Whether node chain testing is enabled (**true**) or not (**false**). The default is **false**.
gcOptions.*	**ocomcCore.ocomc.nodeMgrOptions**	The JVM garbage collection (GC) settings to apply to the Node Manager pods. • **globalGC**: The global JVM GC settings to apply to all Node Manager pods. This value takes precedence over the **gc.***x* keys. • **gc.***x*: The JVM GC settings to apply to the specified Node Manager pod. For example, **gc.1** applies to **node-mgr-app**, **gc.2** applies to **node-mgr-app-2**, and so on.
memoryOptions.*	**ocomcCore.ocomc.nodeMgrOptions**	The global JVM memory settings to apply to all Node Manager pods. • **globalMem**: The global JVM memory settings to apply to all Node Manager pods. This value takes precedence over the **mem.***x* keys. • **mem.***x*: The JVM memory settings to apply to the specified Node Manager pod. For example, **mem.1** applies to **node-mgr-app**, **mem.2** applies to **node-mgr-app-2**, and so on.
gcOptions	**ocomcCore.ocomc.adminSvrOptions**	The JVM GC settings to apply to the Administration Server pod.
memoryOptions	**ocomcCore.ocomc.adminSvrOptions**	The JVM memory settings to apply to the Administration Server pod.
adminserver.type	**ocomcCore.ocomc.service**	The service type for the Administration Server pod: **ClusterIP** or **NodePort**. The default is **ClusterIP**.
nodemgr.type	**ocomcCore.ocomc.service**	The service type for the Node Manager pod: **ClusterIP** or **NodePort**. The default is **ClusterIP**.
type	**ocomcCore.ocomc.service**	The service type for the Administration Server and Node Manager pod: **NodePort** or **ClusterIP**. The default is **ClusterIP**.
name	**ocomcCore.ocomc.storageClass**	The storage class name for persistent volume claims (PVCs).
copyTrustStore	**ocomcCore.ocomc.job**	Set this to **true** before you run any scaling job if SSL is enabled. The default is **false**. For more information, see "Scaling Node Manager Pods".
scaleUpNMDifferentDataPV.*	**ocomcCore.ocomc.job**	Details about the pods to replicate when you scale up the number of Node Manager pods. Add these keys only if your Node Manager pods will have different data PVs: • **flag**: Set this to true if your Node Manager pods use different data PVs. • **parent_NM**: The Node Manager pod to replicate in the format: *name@host***:***port*. • **exportConfigFile**: The file name and absolute path of the node chain configuration file to import, without any extensions.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
scaleUpNMSameDataPV.*	**ocomcCore.ocomc.job**	Details about the pods to replicate when you scale up the number of Node Manager pods. Add these keys only if your Node Manager pods will share the same data PV: • **flag**: Set this to **true** if your Node Manager pods share the same data PV. • **CC_NM**: The Node Manager pod that contains all of your Collection Cartridge (CC) nodes in the format: *name@host***:***port*. This Node Manager must contain only CC nodes. • **EPDC_NM**: The Node Manager pod that contains all of your Enhancement Processor (EP) and Distribution Cartridge (DC) nodes in the format: *name@host***:***port*. This Node Manager must contain only EP and DC nodes. **Note:** List only DC and scalable EP Node Manager pods. That is, do not include Duplicate Check EP Node Managers. • **exportConfigFile**: The file name and absolute path of the node chain configuration file to import, without any extensions. • **importNMList**: The comma-separated list of nodes to import, in the order in which they appear in the node chain. For example: *CCNM_name@CCNM_host***:***Port*, *EPDC_name@EPDC_host***:***Port*.
scaleDownNM.*	**ocomcCore.ocomc.job**	Details about the pod scale down job. • **flag**: Whether to run the scale down job (**true**) or not (**false**). The default is **false**. • **startAllNodes**: Whether to restart all of the cartridges in the Administration Server (**yes**) or not (**no**). The default is **no**.
postScalingDownNM.*	**ocomcCore.ocomc.job**	Details about the post pod scale down job. • **flag**: Whether to run the post scale down job (**true**) or not (**yes**). • **backupNeeded**: Whether to back up the installation to the **/home/ocomcuser/install/hostName_bkp** directory (**yes**) or not (**no**). The default is **no**.

Table 4-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
runNMShell.*	**ocomcCore.ocomc.job**	The details for running the NMShell job: • **flag**: Whether to run the NMShell job (**true**) or not (**false**). • **fileExtension**: The file name extension of all input files to read. Only files with the specified extensions are read. The input file is a list of NMShell commands to run. • **inputDir**: The directory in which the input files will be placed. • **strictMode**: Specifies what happens if an error is encountered while processing a command in the input file. Possible values are **cmd**, **block**, or **no**. The default is **no**. • **hook**: Enter **pre-upgrade** if this job will be run just before a Helm command. Enter **post-upgrade** if this job will be run just before a Helm install. • **hookWeight**: This key applies only when the scaling down job and NMShell job are both enabled. The lowest weight number of the two jobs gets loaded first.
nodemgr.* For Node Manager pods that use different data PVs	**ocomcCore.ocomc.deployment**	Information about the Node Manager pods to create. • **count**: The total number of Node Manager pods to create. Set this to **1** for the first deployment. • **sharedDataPV**: Set this to **false**.
nodemgr.* For Node Manager pods that share the same data PV	**ocomcCore.ocomc.deployment**	Information about the Node Manager pods to create. Use these keys only for Node Manager pods that share the same data PV. • **ccNMRangeEnd**: The ending rage for creating CC Node Manager pods. (The starting range is always 1). • **epdcNMRangeStart**: The starting range for creating the EP and DC Node Manager pods. • **epdcNMRangeEnd**: The ending range for creating the EP and DC Node Manager pods. **Note:** The range must include both scalable and non-scalable EP Node Manager pods. • **sharedDataPV**: Set this to **true**.

Deploying Offline Mediation Controller Services

To deploy Offline Mediation Controller services on your cloud native environment, do this:

> **Note:**
>
> To integrate the Offline Mediation Controller cloud native deployment with the ECE and BRM cloud native deployments, they must use the same namespace.

1. Validate the content of your charts by entering this command from the **helmcharts** directory:

   ```
   helm lint --strict oc-cn-ocomc
   ```

   You'll see this if the command completes successfully:

   ```
   1 chart(s) linted, no failures
   ```

2. Run the **helm install** command from the **helmcharts** directory:

   ```
   helm install ReleaseName oc-cn-ocomc --namespace NameSpace --values
   OverrideValuesFile
   ```

   where:

   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects. To integrate the Offline Mediation Controller cloud native deployment with the ECE and BRM cloud native deployments, they must use the same namespace.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.

   For example, if the **override-values.yaml** file is in the **helmcharts** directory, the command for installing Offline Mediation Controller cloud native services would be:

   ```
   helm install ocomc oc-cn-ocomc --namespace ocgbu --values override-
   values.yaml
   ```

5

About Integrating Offline Mediation Controller REST Services Manager with Cloud Native

You can integrate an external application with Oracle Communications Offline Mediation Controller cloud native by using Offline Mediation Controller REST Services Manager (RSM).

Topics in this document:

About Offline Mediation Controller REST Services Manager

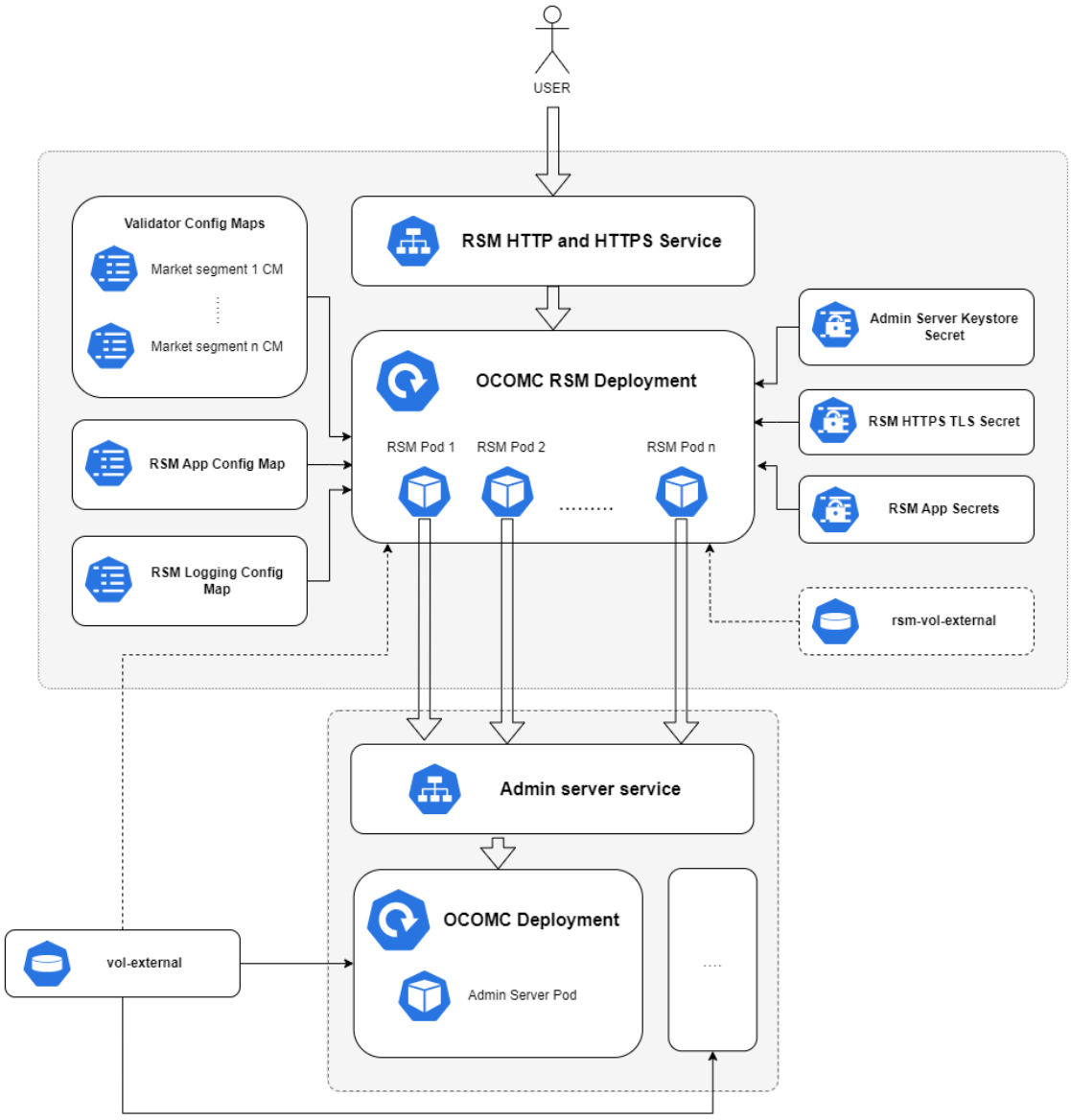
The Offline Mediation Controller REST Services Manager (RSM) allows you to perform the same operations as the **NMShell** application using external client applications. For example, it allows your external application to do the following in Offline Mediation Controller:

- Manage Nodes
- Manage Node Managers
- Retrieve a list of node chains
- Compile and save the NPL rules file
- Export node configurations and customizations

About Offline Mediation Controller REST Services Manager (RSM) Cloud Native Architecture

Figure 5-1 shows all the components of the Offline Mediation Controller REST Services Manager (RSM) cloud native architecture.

Figure 5-1 Offline Mediation Controller REST Services Manager (RSM) Cloud Native Architecture



The components in this figure include:

- **RSM Deployment:** The primary deployment of RSM with all the necessary components and configurations.
- **RSM HTTP and HTTPS Service:** This service exposes RSM to ports, allowing access to RSM through HTTP and HTTPS protocols.
- **Validator ConfigMap:** There is a unique ConfigMap for each market segment.
- **RSM App ConfigMap:** The ConfigMap contains the **application.yaml** file, which holds the configurations required to initiate the RSM server.
- **RSM Logging ConfigMap:** The ConfigMap holds the **log4j2.yaml** file, encompassing logging-related configurations.

- **Admin Server Keystore Secret:** This secret contains the administration server KeyStore file in a Base64-encoded format.
- **RSM HTTPS TLS Secret:** This secret contains the HTTPS TLS store utilized by RSM when the HTTPS protocol is enabled.
- **RSM App Secret:** This secret contains all confidential information necessary to launch the RSM server.
- **vol-external:** This is an optional PV reference. RSM will incorporate it only if the flag **rsm.pvc.ocomcExternal.enabled** in the **override-values.yaml** file is set to **true**. When enabled, the RSM will share the **vol-external** PV of the OCOMC core deployment. It is mandatory to enable this flag if the node chain solution includes cartridges containing sensitive information such as FTP or database passwords.
- **rsm-vol-external**: This PV is optional and can be enabled by setting the flag **rsm.pvc.external.enabled** to **true** in the **values.yaml** file. When enabled, the RSM will load custom cartridges from the specified PV into the classpath. The source directory for it can be configured in the **override-values.yaml** file.

Installing Offline Mediation Controller REST Services Manager

The Offline Mediation Controller REST Services Manager can be installed along with core Offline Mediation Controller components using a unified Helm chart.

To install Offline Mediation Controller REST Services:

1. Configure and install all required third-party software. See "Setting Up Prerequisite Software".
2. Configure the Offline Mediation Controller server and REST Services Manager connection. See "Configuring the Offline Mediation Controller Core and REST Services Manager Connection".
3. Configure the RSM server. See "Configuring the REST Services Manager (RSM) Server".
4. Load custom validators. See "Configuring and Loading Custom Validators".
5. Deploy Offline Mediation Controller REST Services Manager. See "Deploying Offline Mediation Controller Services".

Setting Up Prerequisite Software

As part of preparing your environment for Offline Mediation Controller RSM, you install and set up various components and services in ways that are best suited for your cloud native environment. The following shows the high-level prerequisite tasks for deploying Offline Mediation Controller RSM:

1. Ensure that you have downloaded the latest software that is compatible with Offline Mediation Controller cloud native. See "Offline Mediation Controller Cloud Native System Requirements" in *Offline Mediation Controller Compatibility Matrix.*
2. Ensure that your environment setup is complete. See "Setting Up Your Environment".
3. Download the Offline Mediation Controller cloud native Helm chart. See "Downloading Packages for the Offline Mediation Controller Cloud Native Helm Charts".

Configuring the Offline Mediation Controller Core and REST Services Manager Connection

To configure the Offline Mediation Controller core and REST Services Manager connection:

1. In your **override-values.yaml** file for **oc-cn-ocomc**, set the following keys:
   - **ocomcRSM.rsm.adminServerConnection.hostname**: Specify the hostname where the Offline Mediation Controller Admin Server is running.
   - **ocomcRSM.rsm.adminServerConnection.port**: Specify the port where Offline Mediation Controller Admin Server listens.
2. If Offline Mediation Controller core uses SSL, do the following:
   a. Copy your **adminClientTruststore.jks** file from the **vol-keystore PV** of Offline Mediation Controller core to the **oc-cn-ocomc/rsm/ocomc-rsm-keystore** directory.
   b. In your **override-values.yaml** file, set the following keys:
      - **ocomcRSM.rsm.adminServerConnection.ssl.enabled**: Set this key to **true**. This enables SSL between REST Services Manager and the Admin Server.
      - **ocomcRSM.rsm.adminServerConnection.ssl.keystoreName**: Specify the name of your KeyStore file, such as **adminClientTruststore.jks**.
3. If authentication is enabled for Offline Mediation Controller core, set the following keys in your **override-values.yaml** file:
   - **ocomcRSM.rsm.adminServerConnection.username**: Specify the user name for logging in to the Admin Server.
   - **ocomcRSM.rsm.adminServerConnection.password:** Specify the password for logging in to the Admin Server.

Configuring the REST Services Manager (RSM) Server

To configure the Offline Mediation Controller RSM Server:

1. Enable HTTPS in RSM by doing the following:
   a. Copy your generated **.p12** KeyStore file to the RSM Helm chart directory (**oc-cn-ocomc/charts/oc-cn-ocomc-rsm/ocomc-rsm-keystore**).
   b. Set the following keys in your **override-values.yaml** file for **oc-cn-ocomc:**
      - **ocomcRSM.rsm.https.enabled**: Set this to **true**.
      - **ocomcRSM.rsm.https.ketsotreName**: Specify the name of the KeyStore file with the extension.
      - **ocomcRSM.rsm.https.keystorePassPhrase**: Specify the KeyStore passphrase.
2. Expose RSM through a NodePort by setting the following keys in your **override-values.yaml** file:
   - **ocomcRSM.rsm.service.type**: Set this to **NodePort**.
   - **ocomcRSM.rsm.service.nodePort**: Specify the port number.
   - **ocomcRSM.rsm.https.service.nodePort**: If the HTTPS port is enabled, specify the port for exposing the HTTPS port outside the cluster.

3. Enable Oracle Access Management Auth by setting the following keys in your **override-values.yaml** file:
   a. **ocomcRSM.rsm.security.provider**: Set this to OAM.
   b. **ocomcRSM.rsm.security.configuration.oam**: Fill in the Oracle Access Management and Oracle Unified Directory configuration details.
4. Set the log levels to the appropriate level in the **ocomcRSM.rsm.logging.packagingLogging** keys in your **override-values.yaml** file.

Configuring and Loading Custom Validators

In Offline Mediation Controller REST Services Manager, you can configure custom validators.

To load custom validators:

1. Enable custom validators for Offline Mediation Controller RSM. In your **override-values.yaml** file for **oc-cn-ocomc**, set the **ocomcRSM.rsm.customisation.nodeConfigValidator.validators.enabled** key to **true**.
2. Create a subdirectory within the RSM Helm chart directory (**oc-cn-ocomc-rsm/ocomc-rsm-validator**) with the name of the market segment for the validator. For example, create a directory named **oc-cn-ocomc-rsm/ocomc-rsm-validator/my-market**.
3. Copy the validator YAML files into the directory created in the previous step.
4. In your **override-values.yaml** file, set the **ocomcRSM.rsm.customisation.nodeConfigValidator.validators.marketSegments** key to a list of supported market segments.

About the Offline Mediation Controller REST Services Manager Keys

Table 5-1 lists the keys that directly impact Offline Mediation Controller REST Services Manager. Add these keys to your **override-values.yaml** file with the same path hierarchy.

Table 5-1 Offline Mediation Controller REST Services Manager Keys

Key	Path in values.yaml file	Description
imagePullSecrets	-	The location of your **imagePullSecrets**, which stores the credentials (or Secret) for accessing your private Docker registry.
name	**ocomcRSM.rsm**	The name to use for the deployment. The final name of the deployment is derived using the name provided.
fullname	**ocomcRSM.rsm**	The final name of the deployment to use. This would be used for the deployment without any modification.
replicas	**ocomcRSM.rsm**	The total number of RSM pods to run in the deployment.
restartCount	**ocomcRSM.rsm**	Tracks the number of restarts. To restart the pods, increment the value by 1 and run the **helm upgrade** command.
annotations	**ocomcRSM.rsm**	The custom annotations to add to the deployment (should be a key-value pair).
additionalLabels	**ocomcRSM.rsm**	The custom labels to add to the deployment (should be a key-value pair).

Table 5-1 (Cont.) Offline Mediation Controller REST Services Manager Keys

Key	Path in values.yaml file	Description
nodeSelectors	**ocomcRSM.rsm**	The node selector to use for the deployment.
imageRepository	**ocomcRSM.rsm.container**	The repository from where the RSM image can be pulled. **Note:** The repository URI should not end with a trailing slash.
imagePullPolicy	**ocomcRSM.rsm.container**	The image pull policy to use for the deployment. The default value is **IfNotPresent**, which specifies not to pull the image if it's already present. Applicable values are **IfNotPresent** and **Always**.
image	**ocomcRSM.rsm.container**	The RSM image name and tag concatenated with a colon (**:**). Ensure to align with the RSM image version to be deployed.
enabled	**ocomcRSM.rsm.https**	Whether RSM should run with HTTPS.
keystoreName	**ocomcRSM.rsm.https**	The KeyStore file name with its extension to use for HTTPS. The file must be present in the **oc-cn-ocomc-rsm/ocomc-rsm-keystore** directory.
keystorePassPhrase	**ocomcRSM.rsm.https**	The passphrase for the HTTPS KeyStore file.
service.nodePort	**ocomcRSM.rsm.https**	The node port to use for HTTPS service. This would be used when the service type of RSM is set to NodePort.
hostname	**ocomcRSM.rsm.adminServerConnection**	The host name for accessing the admin server.
port	**ocomcRSM.rsm.adminServerConnection**	The port at which the admin server is listening on.
username	**ocomcRSM.rsm.adminServerConnection**	The user name to use for logging into the admin server.
password	**ocomcRSM.rsm.adminServerConnection**	The password for the specified user to use during login.
ssl.enabled	**ocomcRSM.rsm.adminServerConnection**	Whether Offline Mediation Controller core is SSL enabled.
ssl.keystoreName	**ocomcRSM.rsm.adminServerConnection**	If Offline Mediation Controller Core is SSL enabled, specify the admin-server TrustStore file name, including its extension, to use during the connection to the admin server. The file must be present in the **oc-cn-ocomc-rsm/ocomc-rsm-keystore** directory. By default, the TrustStore in Offline Mediation Controller core can be copied from *OMC_home*/**config/GUI/adminClientTruststore.jks**.
testNodeChain.enabled	**ocomcRSM.rsm**	Whether node chain testing is enabled (**true**) or not (**false**). The default is **false**.
ocomcExternal.enabled	**ocomcRSM.rsm.pvc**	Whether RSM shares the same external PV of the Offline Mediation Controller core. Enabling this is mandatory when RSM is involved in creating node chain solutions involving cartridges with sensitive password information (FTP or database passwords). The mount path is **/app/volumes/ocomc-ext**.
ocomcExternal.name	**ocomcRSM.rsm.pvc**	The name of the external volume in Offline Mediation Controller Core.
external.enabled	**ocomcRSM.rsm.pvc**	Whether to create an external PV for RSM. The mount path is **/app/volumes/ext**.
external.name	**ocomcRSM.rsm.pvc**	The name for RSM's external volume.

Table 5-1 (Cont.) Offline Mediation Controller REST Services Manager Keys

Key	Path in values.yaml file	Description
external.accessModes	**ocomcRSM.rsm.pvc**	The access mode to use for RSM's external PV.
external.storage	**ocomcRSM.rsm.pvc**	The storage to use for the RSM's external PV.
name	**ocomcRSM.rsm.storageClass**	The storage class to use if RSM's external PV is enabled.
cartridgeFolder	**ocomcRSM.rsm.configEnv**	The directory path where RSM retrieves and loads cartridges from.
nodeTypeMapper.enabled	**ocomcRSM.rsm.customisation**	Whether to load custom nodeMappers into RSM. The content of the file needs to be added to **oc-cn-ocomc-rsm/templates/configmap-nodetypemapper.yaml.**
nodeConfigValidator.enableCache	**ocomcRSM.rsm.customisation**	Whether caching should be enabled to the Node Config Validator.
nodeConfigValidator.validators.enabled	**ocomcRSM.rsm.customisation**	Whether to load custom validators into RSM.
nodeConfigValidator.validators.marketSegments	**ocomcRSM.rsm.customisation**	The list of market segments of the validators to load. The name must map to a directory in the **oc-cn-ocomc-rsm/ocomc-rsm-validator** directory.
nodeTypeMetadata.enabled	**ocomcRSM.rsm.customisation**	Whether to load custom node type metadata files into RSM.
nodeTypeMetadata.marketSegments	**ocomcRSM.rsm.customisation**	The list of market segments of the metadata node to load. The name must map to a directory in the **oc-cn-ocomc-rsm/ocomc-rsm-metadata** directory.
additional labels	**ocomcRSM.rsm.service**	The custom labels to add for the service.
additionalAnnotation	**ocomcRSM.rsm.service**	The custom annotations to add for the deployment.
service.type	**ocomcRSM.rsm.service**	The Kubernetes service type to use.
nodePort	**ocomcRSM.rsm.service**	The NodePort that RSM should be exposed to, if service type is set to **NodePort**.
provider	**ocomcRSM.rsm.security**	The security provider for user authentication.
clientId	**ocomcRSM.rsm.security.configuration.oam**	The Oracle Access Management (OAM) application client ID.
clientSecret	**ocomcRSM.rsm.security.configuration.oam**	The OAM application client secret.
tokenEndpointUri	**ocomcRSM.rsm.security.configuration.oam**	The OAM token endpoint URI.
authorizationEndpointUri	**ocomcRSM.rsm.security.configuration.oam**	The OAM authorization endpoint URI.
introspectEndpointUri	**ocomcRSM.rsm.security.configuration.oam**	The OAM introspect endpoint URI.
oauthIdentityDomainName	**ocomcRSM.rsm.security.configuration.oam**	The OAM client's identity domain name.
oudHostName	**ocomcRSM.rsm.security.configuration.oam**	The Oracle Unified Directory (OUD) server's host name.
oudAdminUserName	**ocomcRSM.rsm.security.configuration.oam**	The OUD server's administration user name.
oudAdminUserPassword	**ocomcRSM.rsm.security.configuration.oam**	The OUD server's administration user password.

Table 5-1 (Cont.) Offline Mediation Controller REST Services Manager Keys

Key	Path in values.yaml file	Description
oudHttpPort	**ocomcRSM.rsm.security.configuration.oam**	The OUD HTTP port.
oudHttpsPort	**ocomcRSM.rsm.security.configuration.oam**	The OUD HTTPS port.
oudUsersBaseDn	**ocomcRSM.rsm.security.configuration.oam**	The user's base DN in OUD.
oudGroupsBaseDn	**ocomcRSM.rsm.security.configuration.oam**	The group's base DN in OUD.
jvmOpts	**ocomcRSM.rsm**	The required JVM configuration for RSM.
terminationGracePeriodSeconds	**ocomcRSM.rsm**	The termination grace period for the pod. This is optional.
format.type	**ocomcRSM.rsm.logging**	The logging layout to use. The value should be a supported log4j logging layout.
format.pattern	**ocomcRSM.rsm.logging**	The logging pattern to use.
rootLevel	**ocomcRSM.rsm.logging**	The RSM's root logging level.
packageLogging	**ocomcRSM.rsm.logging**	The logging levels specific to individual packages.

6

Upgrading Offline Mediation Controller

Learn how to upgrade your existing Oracle Communications Offline Mediation Controller cloud native deployment to the latest release.

Topics in this document:

- Upgrading Offline Mediation Controller from 12.0 Patch Set 4 or Later to 15.0
- Upgrading Offline Mediation Controller from 12.0 Patch Set 3 to 15.0

In this document, the Offline Mediation Controller release running on your production system is called the existing release. The release you are upgrading to is called the new release. For example, if you are upgrading from Offline Mediation Controller 12.0 Patch Set 4 to Offline Mediation Controller 15.0, 12.0 Patch Set 4 is the existing release and 15.0 is the new release.

Upgrading Offline Mediation Controller from 12.0 Patch Set 4 or Later to 15.0

When you upgrade your Offline Mediation Controller cloud native services, it upgrades all core services in your Offline Mediation Controller cloud native environment.

To upgrade your Offline Mediation Controller cloud native services from the 12.0 Patch Set 4 or later release to the 15.0 release:

1. Download the Helm charts for the Offline Mediation Controller cloud native deployment. See "Downloading Packages for the Offline Mediation Controller Cloud Native Helm Charts".
2. Download the Offline Mediation Controller cloud native images in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling Offline Mediation Controller Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading Offline Mediation Controller Images from Oracle Software Delivery Website".
3. Extract the Offline Mediation Controller Helm chart from the archive:

   ```
   tar xvzf oc-cn-ocomc-15.0.0.x.0.tgz
   ```

   where *x* is **0** for the 15.0.0.0.0 release, **1** for Offline Mediation Controller 15.0 Patch Set 1, **2** for Offline Mediation Controller 15.0 Patch Set 2, and so on. If you are extracting an interim patch, the file name will also have the interim patch number appended to it, such as oc-cn-ocomc-15.0.0.*x*.0-12345678.tgz.
4. Set the following keys in your **override-values.yaml** file for **oc-cn-ocomc**:
   - **charts.enableCore**: Set this to **true**.
   - **charts.enableRSM**: Set this to **true** if you want to use Offline Mediation Controller REST Services Manager in your 15.0 release.
   - **ocomcCore.upgrade.fromRelease**: Set this to the Offline Mediation Controller release number you are upgrading from. For example, if you are upgrading from 12.0 Patch Set 5, set the key to **12.0.0.5.0**.

- **ocomcCore.ocomc.configEnv.forceGenSslcert**: Set this to **true** if SSL is enabled in your system.
- Set the other keys in Table 4-2 as needed.

5. Run the **helm upgrade** command from the **oc-cn-ocomc** directory:

   ```
   helm upgrade NameSpace oc-cn-ocomc -values OverrideValuesFile -n
   ReleaseName
   ```

   where:

   - *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.
   - *ReleaseName* is the release name, which is used to track this installation instance.

6. In your **override-values.yaml** file for **oc-cn-ocomc**, reset the **ocomcCore.ocomc.configEnv.forceGenSslcert** key to **false**.

> **Note:**
>
> If there are config keys present in the sub-charts, or from previous installations during upgrades, which are not found in the new **values.yaml** file, users can easily add it to the **override-values.yaml** file.

Upgrading Offline Mediation Controller from 12.0 Patch Set 3 to 15.0

The process for upgrading Offline Mediation Controller cloud native from the 12.0 Patch Set 3 release to the 15.0 release is a two-step process:

1. Upgrade your Offline Mediation Controller cloud native system from 12.0 Patch Set 3 to 12.0 Patch Set 4.
2. Upgrade your Offline Mediation Controller cloud native system from 12.0 Patch Set 4 to 15.0 by following the instructions in "Upgrading Offline Mediation Controller from 12.0 Patch Set 4 or Later to 15.0".

To upgrade your Offline Mediation Controller cloud native services to the 12.0 Patch Set 4 release:

1. Download Oracle Communications Offline Mediation Controller 12.0 Patch Set 4 from the Oracle Support website (https://support.oracle.com).
2. Unzip the file. The Zip file includes the **oc-cn-ocomc-helm-chart-12.0.0.4.0** archive file for upgrading your system.
3. Extract the Offline Mediation Controller Helm chart from the archive:

   ```
   tar xvzf oc-cn-ocomc-helm-chart-12.0.0.4.0.tgz
   ```

4. In admin-server-pvc, rename your 12.0 Patch Set 3 *OMC_home***/config/adminserver/systemModel.cfg** file to **SystemModel.orig**.

5. Move your 12.0 Patch Set 3 wallet files from their existing location (vol-keystore PV) to the vol-external PV.
6. Set the following keys in your 12.0 Patch Set 4 **override-values.yaml** file:
   - **ocomcSoftwareUpgradePath**: Set this to **/container-scripts/OCOMC-12.0.0.4.0_generic_full.jar**.
   - **ocomc.job.preUpgradeToPS4.flag**: Set this to **true**.
   - **upgrade.fromRelease**: Set this to **12.0.0.3.0**.
   - **ocomc.configEnv.forceGenSslcert**: Set this to **true**.
   - **ocomc.deployment.nodemgr.count**: Ensure this is set to **1**.
7. Run the **helmUpgradeToPS4.sh** script from the **oc-cn-ocomc-helm-chart** directory:

   ```
   sh helmUpgradeToPS4 NameSpace OverrideValuesFile ReleaseName LogFile
   ```

   where:
   - *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.
   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *LogFile* is the path and name to the log file to create.

   Wait for the admin-server-app pod to start.
8. Set the following keys in your 12.0 Patch Set 4 **override-values.yaml** file:
   - **ocomc.job.preUpgradeToPS4.flag**: Set this to **false**.
   - **ocomc.job.postUpgradeToPS4.flag**: Set this to **true**.
   - **ocomc.job.postUpgradeToPS4.existingNM**: Set this to the name of the existing Node Manager pod in the format *hostname@host*:*port*, where *name* is the mediation host's name configured in Node Manager, *host* is the IP address or hostname of the server on which the mediation host resides, and *port* is the port number at which the mediation host communicates with the Node Manager pod.
9. Run the **helm upgrade** command:

   ```
   helm upgrade ReleaseName oc-cn-ocomc-helm-chart --values
   OverridingValueFile -n Namespace
   ```

10. Because the wallet location changed in the 12.0 Patch Set 4 release, update all nodes that use the wallet to point to the correct wallet location.
11. Set the following keys in your **override-values.yaml** file:
    - **ocomc.job.postUpgradeToPS4.flag**: Set this to **false**.
    - **ocomc.configEnv.forceGenSslcert**: Set this to **false**.

7

Connecting Your Administration Client

Learn how to connect your Oracle Communications Offline Mediation Controller cloud native deployment with an on-premises version of Offline Mediation Controller Administration Client.

Topics in this document:

About Administration Client

Administration Client is a GUI application that you use for creating node chains and editing rule files. You also use Administration Client for administrating Offline Mediation Controller. For example, you can use it to manage users and define instances of system components.

For more information about using Administration Client, see "About Configuring Nodes and Node Chains" in *Offline Mediation Controller User's Guide*.

Connecting Administration Client

Although Offline Mediation Controller can be deployed on a cloud native environment, you must install an on-premise version of Administration Client to work with it.

To set up a connection between your on-premises Administration Client and the Administration Server on a cloud native environment, do the following:

1. Configure the Administration Server cloud native service to connect to your Administration Client. See "Configuring Administration Server Cloud Native".
2. Install an on-premises version of Administration Client on one of the following:
   - On a physical server that is reachable to the Kubernetes node where the Administration Server pod is running.
   - If graphical desktop support such as VNC is available on a worker node, you can install Administration Client on the same worker node in which the Administration Server and Node Manager pods are running.

   See "Installing Offline Mediation Controller Administration Client" in *Offline Mediation Controller Installation Guide*.
3. Perform postinstallation tasks on the Administration Client machine. See "Postinstallation Tasks for Administration Client".
4. Verify that your Administration Client can connect to the Administration Server. See "Verifying the Administration Client Connection".

After your Administration Client has connected successfully, ensure that you place all CDR files inside the vol-data PVC and that all CDRs have read and write permission for the **ocomcuser** user.

Configuring Administration Server Cloud Native

When configuring your Offline Mediation Controller Administration Server cloud native service, ensure that you do the following:

1. Expose the Administration Server pod (admin-server-app):
   - If your Administration Client is located remotely or is on a Windows system, set the Administration Server's service type to NodePort.
   - If your Administration Client is installed on the same worker node in which the Administration Server pod is running, set the Administration Server's service type to clusterIP.
2. Open your **override-values.yaml** file for **oc-cn-ocomc**.
3. Set the **ocomcCore.ocomc.configEnv.adminsvrIp** key to **admin-server-app**.
4. If your Administration Client is installed remotely or on a Windows system, set these additional keys:
   - **ocomcCore.ocomc.configEnv.nmExternalPort**: Set this to the external port for the Node Manager. Set this key only if your admin-server pod and node-mgr pod are running on different machines.
   - **ocomcCore.ocomc.configEnv.adminsvrExternalPort**: Set this to the external port for the Administration Server.
   - **ocomcCore.ocomc.configEnv.adminsvrFirewallPort**: Set this to the Administration Server firewall port.
5. Save and close your **override-values.yaml** file.
6. Deploy Offline Mediation Controller by following the instructions in "Deploying Offline Mediation Controller Services".

The following shows sample **override-values.yaml** entries for an Administration Client that is installed remotely or on a Windows system:

```
ocomcCore:
ocomc:
  configEnv:
    adminsvrExternalPort: 31000
    adminsvrFirewallPort: 32000
    adminsvrIp: admin-server-app
```

Postinstallation Tasks for Administration Client

After you install Administration Client, perform the following postinstallation tasks:

1. Copy all Offline Mediation Controller cartridges and your custom cartridges from the cloud native environment's **/home/ocomcuser/ext/cartridges** directory to the Administration Client's *OMC_home*/**cartridges** directory.

2. In the Administration Client machine's **/etc/hosts** file, add the IP address of the Kubernetes node where Administration Server is running. For example:

```
IPAddress       Hostname
198.51.100.1    myhost.example.com
```

3. In your Administration Client, specify the location of the Offline Mediation Controller wallet.
   a. Go to the *OMC_home***/bin/** directory and open either the **UDCEnvironment.bat** file (Windows) or the **UDCEnvironment** file (UNIX).
   b. Set the **OCOMC_WALLET_LOCATION** parameter to the externally mounted wallet PV.
   c. Save and close the file.
   d. Restart Offline Mediation Controller. See "Starting Offline Mediation Controller" in *Offline Mediation Controller Installation Guide*.
4. Ensure the Offline Mediation Controller wallet files in the Kubernetes node are accessible to the Administration Client machine.
5. If SSL is enabled, copy the **adminClientTruststore.jks** file from **vol-external PVC** on the cloud native environment to the Administration Client's *OMC_home***/config/GUI** directory.

Verifying the Administration Client Connection

Start your Administration Client to verify it can connect to your Administration Server on the cloud native environment.

To verify the Administration Client connection:

1. Start Administration Client.
   a. Go to the *OMC_home***/bin** directory.
   b. Run the following command:

```
./gui -f
```

   Administration Client starts in the foreground.
2. In the Welcome to Oracle Communications Offline Mediation Controller dialog box, do the following:
   - In the **Host** field, enter **admin-server-app**.
   - In the **Port** field, enter the Administration Server node port number.
   - Enter your user name and password.
3. Click **Connect**.

   If the Administration Client successfully connects to the Administration Server, you will see the Offline Mediation Controller Administration Client window.

8

Enabling TLS 1.3 Support in Offline Mediation Controller (Release 15.0.1 or later)

This section provides instructions for enabling TLS 1.3 support in Offline Mediation Controller deployments, enhancing communication security. TLS 1.3 offers improved security features compared to older protocols.

About TLS 1.3 Compatibility

Before enabling TLS 1.3, it is important to understand some potential compatibility considerations. When it comes to backwards compatibility, TLS 1.3 can negotiate with older clients (TLS 1.2 and below) but has some key differences:

- TLS 1.3 uses a half-close policy, while TLS 1.2 and above earlier use a duplex-close policy. Applications that depend on the latter duplex-close policy may encounter compatibility issues when upgrading to TLS 1.3.
- The signature_algorithms_cert extension warrants the use of pre-defined signature algorithms for certificate authentication.
- The DSA signature algorithm is not supported in TLS 1.3. A server cannot negotiate with a TLS 1.3 connection if it is configured to only use DSA certificates.
- The supported cipher suites for TLS 1.3 are not the same for TLS 1.2 and earlier versions. Applications with hardcoded cipher suites that are no longer supported may not be able to use TLS 1.3 without modifications to its code.
- Session resumption and key update behaviours are different for TLS 1.3 and TLS 1.2. Although the compatibility impact should be minimal, it is a potential risk if an application depends on the handshake details of the TLS protocols.

Enabling TLS 1.3 Support Automatically

To enable support for TLS 1.3 automatically for your Offline Mediation Controller cloud native deployment:

1. Ensure you are using the latest Offline Mediation Controller 15.0.1.0 image.
2. In your **override-values.yaml** file, set the **ocomcCore.configEnv.forceGenSslcert** flag to **true.**
3. Run the **helm upgrade** command for **oc-cn-ocomc-15.0.1.0.0**.

These steps automatically generate new certificates in the Offline Mediation Controller image using the latest JDK available. If you encounter compatibility issues, enable TLS 1.3 support manually.

Manually Enabling TLS 1.3 Support

To manually enable TLS 1.3 in your Offline Mediation Controller cloud native deployment:

1. Generate a new KeyStore using the **keytool** utility. If generating externally, use the latest Java version.
2. Use a signature algorithm supported by TLS 1.3 during certificate generation.
3. Load the newly generated KeyStore into the appropriate TrustStore.
4. Restart all Offline Mediation Controller components after loading the new keystore.

9

Uninstalling Your Offline Mediation Controller Cloud Native Deployment

Learn how to uninstall your Oracle Communications Offline Mediation Controller cloud native deployment.

Topics in this document:

Uninstalling Your Offline Mediation Controller Cloud Native Deployment

When you uninstall a Helm chart from your Offline Mediation Controller cloud native deployment, it removes only the Kubernetes objects that it created during installation.

Before you uninstall the Offline Mediation Controller Helm chart, back up all data inside mounted file systems.

To uninstall, enter this command:

```
helm delete ReleaseName -n Namespace
```

where:

- *ReleaseName* is the name you assigned to this installation instance.
- *NameSpace* is the namespace in which the Offline Mediation Controller Kubernetes objects reside.

10

Monitoring and Maintaining Offline Mediation Controller Cloud Native

Learn how to maintain your Oracle Communications Offline Mediation Controller cloud native deployment.

Topics in this document:

Using Prometheus Operator to Monitor Offline Mediation Controller Cloud Native

Offline Mediation Controller cloud native tracks and exposes the following metric data in Prometheus format:

- Node Manager-level statistics, which include:
  - The total network account records (NARs) processed
  - The current NARs processed
  - The current processing rate
  - The average processing rate

  Node Manager-level statistics are exposed through the endpoint **http://***hostname***:8082/metrics**, where *hostname* is the host name of the machine on which Offline Mediation Controller cloud native is running. You can change the port number where the metric data is exposed using the **ocomcCore.ocomc.configEnv.metricsPort** key in your **override-values.yaml** file for **oc-cn-ocomc**.
- JVM metrics for all Offline Mediation Controller components, which include:
  - Performance on the Node Manager level
  - JVM parameters

  JVM metrics are exposed through the endpoint **http://***hostname***:***portJVM***/metrics**, where *portJVM* is the port number where the JVM metrics are exposed. You can set the port number by using the **ocomcCore.ocomc.configEnv.metricsPortCN** key in your **override-values.yaml** file for **oc-cn-ocomc**.

To monitor Offline Mediation Controller more easily, you can configure an external centralized metrics service, such as Prometheus Operator, to scrape metrics from each endpoint and store them for analysis and monitoring. You can then set up a visualization tool, such as Grafana, to display your metric data in a graphical format.

For the list of compatible Prometheus Operator and Grafana software versions, see "Offline Mediation Controller Cloud Native Deployment Software Compatibility" in *Offline Mediation Controller Compatibility Matrix*.

Enabling the Automatic Scraping of Metrics

You can configure the Prometheus Operator ServiceMonitor to automatically scrape Offline Mediation Controller metrics. For more information about Prometheus Operator and ServiceMonitors, see the prometheus-operator documentation on the GitHub website (https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/user-guides/getting-started.md).

To enable the automatic scraping of Offline Mediation Controller metrics:

1. Install Prometheus Operator on your cloud native environment.
2. In your **override-values.yaml** file for **oc-cn-ocomc**, set the **serviceMonitor.enabled** key to **true**:

```
ocomc:
   serviceMonitor:
      enabled: true
```

3. Run the **helm upgrade** command to update the Offline Mediation Controller Helm release:

```
helm upgrade ReleaseName oc-cn-ocomc --values OverridingValueFile -n Namespace
```

where:

- *ReleaseName* is the release name, which is used to track the installation instance.
- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.
- *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects.

Using the Sample Grafana Dashboards

The Offline Mediation Controller package includes sample Grafana Dashboard templates that you can use for visualizing metrics. To use the sample dashboards, import the following JSON files from the *OMC_home*/**sampleData/dashboards** directory into Grafana:

- **OCOMC_JVM_Dashboard.json**: This dashboard lets you view JVM-related metrics for Offline Mediation Controller.
- **OCOMC_Node_Manager_Summary.json**: This dashboard lets you view NAR processing metrics for the Node Manager.
- **OCOMC_Node_Summary.json**: This dashboard lets you view NAR processing metrics for all nodes.
- **OCOMC_Summary_Dashboard.json**: This dashboard lets you view NAR-related metrics for all Offline Mediation Controller components.

For information about importing dashboards, see "Manage Dashboards" in the *Grafana Dashboards* documentation.

Using NMShell to Automate Deployment of Node Chains

You can use the Offline Mediation Controller Shell (NMShell) tool to:

- Access Offline Mediation Controller cloud native system information
- Make node configuration changes
- Discover the status of one or more nodes
- Perform start and stop operations, basic alarm monitoring, and node configuration changes

For more information, see "Using the Offline Mediation Controller Shell Tool" and "Managing Nodes Using NMShell Command-Line Components" in *Offline Mediation Controller System Administrator's Guide*.

You can run the NMShell tool outside of an Offline Mediation Controller pod by creating an NMShell job. The job runs as a post-upgrade or postinstall hook and executes the NMShell tool using the commands that are passed in input files. You specify the location of your input files and how to handle errors that occur while processing an input file by using keys in your **override-values.yaml** file for **oc-cn-ocomc**.

To create an NMShell job:

1. Create one or more input files that specify the list of NMShell commands to run as part of the NMShell job. Ensure that you add the appropriate extension to the input file name.

   For example input file content, see "Example: Input File for Adding a Mediation Host" and "Example: Input File with Command Blocks".

2. Do one of the following:

   - If the job will run as part of a postinstall hook, create a Dockerfile that copies your input files to an accessible location. For example:

     ```
     FROM oc-cn-ocomc:15.0.0.x.0
     RUN mkdir -p /home/ocomcuser/nmshell
     COPY test.nmshell /home/ocomcuser/nmshell
     COPY export_20220311_024702.nmx /home/ocomcuser/nmshell
     COPY export_20220311_024702.xml /home/ocomcuser/nmshell
     ```

   - If you will run the job as part of a post-upgrade hook, copy your input files to the PV that can be accessed by the Administration Server pod (vol-external PVC).

3. In your **override-values.yaml** file for **oc-cn-ocomc**, set the following keys under **ocomcCore.ocomc.job.runNMShell**:

   - **flag**: Set this to **true**.
   - **fileExtension**: Set this to the file name extension of all input files that you want processed, such as **.input**.
   - **inputDir**: Set this to the absolute path where input files will be placed, such as **/home/ocomcuser/nmshell**.
   - **strictMode**: Specify how to handle errors that occur while processing an input file by setting this key to one of the following:
     - **cmd**: The tool checks the return code after every NMShell command. If a command fails, the NMShell job stops processing that input file. It then starts processing the next input file in the directory.
     - **block**: The tool checks the return code after NMShell processes a command block, which is indicated by **EOB** commands. If a block fails, the NMShell job stops processing that input file. It then starts processing the next input file in the directory.
     - **no**: The tool processes all input files irrespective of their return codes.

- **hook**: Set this key to one of the following:
  - **post-upgrade**: Specifies to run this job as a post-upgrade hook.
  - **postinstall**: Specifies to run this job as postinstall hook.
- **hookWeight**: This key applies only when the scaling down job and NMShell job are both enabled. The lowest number of the two jobs gets loaded first. If you want the NMShell job to run first, set **hookWeight** to a negative number, such as **-1**.

4. Run the NMShell job:
   - To run the job as a postinstall hook, enter this command:

     ```
     helm install ReleaseName oc-cn-ocomc --namespace NameSpace --timeout
     15m --values OverrideValuesFile
     ```

   - To run the NMShell job as a post-upgrade hook, enter this command:

     ```
     helm upgrade ReleaseName oc-cn-ocomc --namespace NameSpace --timeout
     15m --values OverrideValuesFile
     ```

   where:
   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects. To integrate the Offline Mediation Controller cloud native deployment with the ECE and BRM cloud native deployments, they must be set up in the same namespace.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.

After processing each input file, NMShell adds one of these extensions to the input file's name:

- **.err**: An error was encountered while running a command in the input file.
- **.done**: The input file was processed with **strictMode** set to **no**.
- **.success**: All of the commands in the input file were processed successfully.

Example: Input File for Adding a Mediation Host

This shows sample input file content that would do the following:

1. Add a mediation host named Test to node-mgr-app.
2. Change the context to the node-mgr-app Node Manager.
3. List the cartridges in node-mgr-app.

```
addhost -n Test -ip node-mgr-app p 55109
cd node-mgr-app 55109
ls
```

Example: Input File with Command Blocks

This shows sample input file content with blocks of code separated by **EOB** commands. Based on the content, NMShell would do the following:

1. Add a mediation host named Test to node-mgr-app.
2. Import the node customization data from **exportFile.nmx** into the Test mediation host.

3. Import the node configuration data from **exportFile.xml** into the Test mediation host.
4. Start all of the nodes for the currently running mediation host.
5. Add a mediation host named Test1 to node-mgr-app-xyz.
6. List the cartridges in node-mgr-app-xyz.
7. Stop all nodes for the currently running mediation host.

If the **strictMode** key was set to **block**, the NMShell job would check for return codes after processing the last line in each block. That is, it would check for the return code after performing steps 1, 2, 3, 5, and 7. If an error code was returned by one of these steps, the job would stop processing the input file and add the **.err** extension to the input file's name.

```
addhost -n test -ip node-mgr-app p 55109
EOB
import -n test@node-mgr-app:55109 -f exportFile.nmx -c Y
EOB
import -n test@node-mgr-app:55109 -f exportFile.xml -c N
EOB
startNodes
addhost -n test1 -ip node-mgr-app-xyz -p 55109
EOB
ls
stopNodes
EOB
```

Managing a Helm Release

After you install a Helm chart, Kubernetes manages all of its objects and deployments. All pods created through **oc-cn-ocomc** are wrapped in a Kubernetes controller, which creates and manages the pods and performs health checks. For example, if a node fails, a controller can automatically replace a pod by scheduling an identical replacement on a different node.

Administrators can perform these maintenance tasks on a Helm chart release:

- Tracking a Release's Status
- Updating a Release
- Checking a Release's Revision
- Rolling Back a Release to a Previous Revision

Tracking a Release's Status

When you install a Helm chart, it creates a release. A release contains Kubernetes objects, such as ConfigMap, Secret, Deployment, Pod, PersistentVolume, and so on. Not every object is up and running immediately. Some objects have a start delay, but the Helm install command completes immediately.

To track the status of a release and its Kubernetes objects, enter this command:

```
helm status ReleaseName -n Namespace
```

where:

- *ReleaseName* is the name you assigned to this installation instance.

- *NameSpace* is the namespace in which the Offline Mediation Controller Kubernetes objects reside.

Updating a Release

To update any key value after a release has been created, enter this command. This command updates or re-creates the impacted Kubernetes objects, without impacting other objects in the release. It also creates a new revision of the release.

```
helm upgrade ReleaseName oc-cn-ocomc --values OverridingValueFile --values
NewOverridingValueFile -n Namespace
```

where:

- *ReleaseName* is the name you assigned to this installation instance.
- *OverridingValueFile* is the path to the YAML file that overrides the default configurations in the **oc-cn-ocomc/values.yaml** file.
- *NewOverridingValueFile* is the path to the YAML file that has updated values. The values in this file are newer than those defined in **values.yaml** and *OverridingValueFile*.
- *Namespace* is the namespace in which the Offline Mediation Controller Kubernetes objects reside.

Checking a Release's Revision

Helm keeps track of the revisions you make to a release. To check the revision for a particular release, enter this command:

```
helm history ReleaseName -n Namespace
```

where:

- *ReleaseName* is the name you assigned to this installation instance.
- *Namespace* is the namespace in which the Offline Mediation Controller Kubernetes objects reside.

Rolling Back a Release to a Previous Revision

To roll back a release to any previous revision, enter this command:

```
helm rollback ReleaseName RevisionNumber -n Namespace
```

where:

- *ReleaseName* is the name you assigned to this installation instance.
- *RevisionNumber* is the value from the Helm history command.
- *Namespace* is the namespace in which the Offline Mediation Controller Kubernetes objects reside.

Rolling Back an Offline Mediation Controller Cloud Native Upgrade

If you encounter errors after upgrading, you can roll back to a previous version of Offline Mediation Controller.

The following procedure assumes that you have upgraded Offline Mediation Controller from 12.0 Patch Set 5 (Revision 1), to 12.0 Patch Set 6 (Revision 2), and then to 15.0 (Revision 3). To roll back your upgrade from 15.0 to 12.0 Patch Set 6, you would do this:

1. Check the revision history of the Offline Mediation Controller release:

   helm history *ReleaseName* **-n** *Namespace*

   You should see something similar to this:

```
REVISION   UPDATED                   STATUS        CHART APP
VERSION        DESCRIPTION
1          Thu May 30 07:12:46 2030   superseded    oc-cn-ocomc
12.0.0.5.0     Initial install
2          Thu May 30 08:32:09 2030   superseded    oc-cn-ocomc
12.0.0.6.0     Upgraded successfully
3          Thu May 30 09:50:00 2030   deployed      oc-cn-ocomc
15.0.0.0.0     Upgraded successfully
```

2. Roll back the release to Offline Mediation Controller 12.0 Patch Set 6:

   helm rollback *ReleaseName* **2 -n** *BrmNamespace*

   If successful, you will see this:

```
Rollback was a success! Happy Helming!
```

3. Check the revision history of the Offline Mediation Controller release:

   helm history *ReleaseName* **-n** *BrmNamespace*

   If successful, you should see something similar to this:

```
REVISION   UPDATED                   STATUS        CHART APP
VERSION        DESCRIPTION
1          Thu May 30 07:12:46 2030   superseded    oc-cn-ocomc
12.0.0.5.0     Initial install
2          Thu May 30 08:32:09 2030   superseded    oc-cn-ocomc
12.0.0.6.0     Upgraded successfully
3          Thu May 30 09:50:00 2030   superseded    oc-cn-ocomc
15.0.0.0.0     Upgraded successfully
4          Thu May 30 11:25:00 2030   deployed      oc-cn-ocomc
12.0.0.6.0     Roll back to 2
```

Integrating Oracle Unified Directory with Offline Mediation Controller Cloud Native

After verifying the Oracle Unified Directory deployment, follow these steps to integrate it with Offline Mediation Controller cloud native:

1. Open a terminal session within the running pod:

   ```
   kubectl exec -it oud-ds-rs-0 -n oudns -- /bin/bash
   ```

2. Inside the OUD container, create a temporary directory (*tempdir*) to hold configuration files:

   ```
   mkdir /u01/oracle/oud/temp
   ```

3. From the Offline Mediation Controller installation directory, copy the **oudConfig** and **populateDir.ldif** files to *tempdir*:

   ```
   kubectl cp /scratch/username/OCOMC_HOME/bin/oudConfig oudns/oud-ds-rs-0:/u01/
   oracle/oud/temp
   kubectl cp /scratch/username/OCOMC_HOME/bin/populateDir.ldif oudns/oud-ds-rs-0:/u01/
   oracle/oud/temp
   ```

4. Inside *tempdir*, create a file named **populateDirTemp.ldif**. This file updates the user information in Oracle Unified Directory to match the Offline Mediation Controller requirements. Add the following content:

   ```
   dn: uid=Admin,ou=People,dc=ocomcexample.com
   changetype: modify
   replace: userpassword
   userpassword: <adminpassword>
   ```

5. Run the *tempdir**/oudConfig*** script with the Oracle Unified Directory container:

   ```
   sh oudConfig -i <oud_instance_path> -h <oud_host> -p <oud_admin_port> -d
   "oud_binddn" -w <oud_admin_password> -b "dc=ocomcexample.com" -l <oud_ldapport>
   ```

   > **Note:**
   >
   > If the command fails with a "host not found" error, replace the hostname with **localhost** in both the above and below commands.

6. Locate the **values.yaml** file in your Offline Mediation Controller installation directory (e.g., **/scratch/username/ocomc)**. Under the **ocomcCore/lcmc.configEnv** section, add the following fields, adjusting values if necessary to match your setup:

   ```
   adminsvrAuthMode: true

   adminsvrAuthuser: "true"

   adminsvrLdapurl: "ldap://oud-ds-rs-lbr-ldap.oudns.svc.cluster.local:1389"

   oudRootUserDn: cn=Directory Manager

   oudPath: /u01/oracle/user_projects/oud-ds-rs-0/OUD

   oudLdapPort: 1389
   ```

```
oudBaseDn: dc=ocomcexample.com

adminConnectPort: 4444

hostName: oud-ds-rs-0.oud-ds-rs.oudns.svc.cluster.local
```

> **Note:**
>
> You have to upgrade the Offline Mediation Controller helm installation after configuring the **values.yaml** file.

7. Log in to Offline Mediation Controller through Administration Client.

11

Scaling Node Manager Pods

Learn how to scale up and scale down the Node Manager pods in your Oracle Communications Offline Mediation Controller cloud native deployment.

Topics in this document:

Scaling Up Node Manager Pods

You can scale up the number of Node Manager pod replicas in your Offline Mediation Controller cloud native environment based on the pod's CPU or memory utilization. This helps ensure that your Node Manager pods have enough capacity to handle the current traffic demand while still controlling costs.

> **Note:**
>
> If your node chains include duplicate check EP nodes or AP nodes, follow the instructions in "Scaling CC, EP, and DC Nodes without Impacting Non-Scalable Nodes" before you start this procedure.

You scale up Node Manager pods by creating a scale up job, which runs as part of a post-upgrade or postinstall hook.

You scale up Node Manager pods as follows:

1. If you are running the scale up job as part of a post-upgrade hook and the cartridge JARs are not part of the Offline Mediation Controller class path, do the following:
   a. Place the cartridge JARs in the directory specified in the **ocomcCore.ocomc.cofigEnv.cartridgeFolder** key, which can be set to a directory in external-PV.
   b. In your **override-values.yaml** file for **oc-cn-ocomc**, increment the **ocomcCore.ocomc.configEnv.restart_count** key by 1.
   c. Run the **helm upgrade** command to update the Offline Mediation Controller Helm release:

      ```
      helm upgrade ReleaseName oc-cn-ocomc --namespace NameSpace --values
      OverrideValuesFile
      ```

      where:

      - *ReleaseName* is the release name, which is used to track this installation instance.

- *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects. To integrate the Offline Mediation Controller cloud native deployment with the ECE and BRM cloud native deployments, they must be set up in the same namespace.
- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.

All Offline Mediation Controller components are restarted.

2. If you are running the scale up job as part of a postinstall hook and your node chain configuration files include cartridge JARs, do the following:
   a. In your **override-values.yaml** file, set the **ocomcCore.ocomc.cofigEnv.cartridgeFolder** key to **/home/ocomcuser/cartridgeJars/**.
   b. Place the cartridge JARs in the **/home/ocomcuser/cartridgeJars/** directory by creating a Dockerfile similar to the following:

```
FROM oc-cn-ocomc:15.0.0.x.0
RUN mkdir -p /home/ocomcuser/cartridgeJars/
COPY custom_cartridge.jar /home/ocomcuser/cartridgeJars/
```

3. Open your **override-values.yaml** file for **oc-cn-ocomc**.
4. Specify the number of Node Manager pods to create:
   - If your Node Manager pods use different data PVs, set the **ocomcCore.ocomc.deployment.nodemgr.count** key to the desired number of Node Manager pods. For example, to increase the number of pods to 3:

```
ocomc:
   deployment:
      nodemgr:
         count: 3
```

   - If your Node Manager pods use the same data PV, set the starting number and ending number for the range of Collection Cartridge (CC) Node Manager pods to create (**ccNMRangeEnd**). Also, set the starting number and ending number for the range of Enhancement Processor (EP) and Distribution Cartridge (DC) Node Manager pods to create (**epdcNMRangeStart** and **epdcNMRangeEnd**). This range must include both scalable and non-scalable EP Node Manager pods.

     For example:

```
ocomc:
   deployment:
      nodemgr:
         ccNMRangeEnd: 2
         epdcNMRangeStart: 100
         epdcNMRangeEnd: 100
```

     In this case, the following Node Manager pods would be created: node-mgr-app (CC Node Manager), node-mgr-app-2 (CC Node Manager), and node-mgr-app-100 (EP and DC Node Manager).

> **Note:**
>
> – The number ranges for the CC Node Manager pods and the EP and DC Node Manager pods should not overlap.
>
> – The range of EP and DC Node Manager pods to create must include both scalable and non-scalable EP Node Managers.
>
> – The non-scalable EP Node Manager must be the first or last Node Manager pod in the EPDC range.

5. Configure the scale up job:

   - If your Node Manager pods use different data PVs, set the following keys under **ocomc.job**:
     – **scaleUpNMDifferentDataPV.flag**: Set this to **true**.
     – **scaleUpNMDifferentDataPV.parent_NM**: Set this to the Node Manager pod to replicate in the format *Mediation_name@Mediation_host*:*Port*, where *Mediation_name* is the mediation host's name configured in Node Manager, *Mediation_host* is the host name of the server on which the mediation host resides, and *Port* is the port number at which the mediation host communicates with the Node Manager pod.
     – **scaleUpNMDifferentDataPV.exportConfigFile**: Set this key only if you copied node chain configuration files onto the data PV. Set this to the absolute path and file name of your node chain configuration files, but do not include the file name extension.

       For example, if the files are named **export.nmx** and **export.xml**, and they reside in **/home/ocomcuser/ext**, you would set **exportConfigFile** to **/home/ocomcuser/ext/export**.
   - If your Node Manager pods use the same data PV, set the following keys under **ocomc.job**:
     – **scaleUpNMSameDataPV.flag**: Set this to **true**.
     – **scaleUpNMSameDataPV.CC_NM**: Set this to the Node Manager that contains all of your Collection Cartridge (CC) nodes in the format *CCNM_name@CCNM_host*:*Port*, where *CCNM_name* is the mediation host's name configured in Node Manager, *CCNM_host* is the host name of the server on which the mediation host resides, and *Port* is the port number at which the mediation host communicates with the CC Node Manager pod.
     – **scaleUpNMSameDataPV.EPDC_NM**: Set this to the Node Manager that contains all of your Enhancement Processor (EP) and Distribution Cartridge (DC) nodes in the format *EPDC_name@EPDC_host*:*Port*, where *EPDC_name* is the mediation host's name configured in Node Manager, *EPDC_host* is the host name of the server on which the mediation host resides, and *Port* is the port number at which the mediation host communicates with the EP and DC Node Manager pods.

     > **Note:**
     >
     > List only Node Managers with scalable nodes. That is, don't list any Node Managers with Duplicate Check EP nodes or AP nodes.

- **scaleUpNMSameDataPV.exportConfigFile**: Set this key only if you copied node chain configuration files onto the data PV. Set this to the absolute path and file name of your node chain configuration files, but do not include the file name extension. For example: **/home/ocomcuser/customFiles/**.

  For example, if the files are named **export.nmx** and **export.xml**, and they reside in **/home/ocomcuser/ext**, you would set **exportConfigFile** to **/home/ocomcuser/ext/export**.

- **scaleUpNMSameDataPV.importNMList**: Set this to a comma-separated list of Node Manager pods to import, in the order in which they appear in the node chain. For example: *CCNM_name@CCNM_host***:***Port*, *EPDC_name@EPDC_host***:***Port*.

> **Note:**
>
> The Node Manager pods must be listed in the order in which they appear in the node chain.

6. Save and close the file.
7. If you are running the scale up job as a postinstall hook, do the following:
   a. Create a Dockerfile that is similar to the following:

```
FROM oc-cn-ocomc:15.0.0.x.0
RUN mkdir -p /home/ocomcuser/customFiles/
COPY export_20210311_024702.xml /home/ocomcuser/customFiles
COPY export_20210311_024702.nmx /home/ocomcuser/customFiles
```

   b. Run the **helm install** command:

```
helm install ReleaseName oc-cn-ocomc --namespace NameSpace --values
OverrideValuesFile --timeout 15m
```

   Before scaling up the pods, the job confirms that the desired Node Manager pods are up and running, that a connection with the Administration Server has been established, and that all Node Manager hosts are reachable.
8. If you are running the scale up job as a post-upgrade hook, run the **helm upgrade** command:

```
helm upgrade ReleaseName oc-cn-ocomc --namespace NameSpace --values
OverrideValuesFile --timeout 15m
```

   Before scaling up the pods, the job confirms that the desired Node Manager pods are up and running, that a connection with the Administration Server has been established, and that all Node Manager hosts are reachable.
9. In your **override-values.yaml** file for **oc-cn-ocomc**, set the **ocomcCore.ocomc.job.scaleUpNMDifferentDataPV.flag** or **ocomcCore.ocomc.job.scaleUpNMSameDataPV.flag** key to **false** so that the jobs are not run again the next time you update the Helm release.

You can check the job's status in one of these log files:

- *OMC_home***/log/scaleUpNMSegregatedDataPV-***Date***.log**

- *OMC_home*/**log/scaleUpNMSameDataPV-***Date*.**log**

Scaling Down Node Manager Pods

> **Note:**
>
> Scaling down Node Manager pods is supported in Offline Mediation Controller 15.0.

You scale down Node Manager pods in your Offline Mediation Controller cloud native environment by removing the Node Manager from the Administration Server, bringing down the Node Manager pods from the Kubernetes cluster, and then cleaning up the Offline Mediation Controller installation.

> **Note:**
>
> If your node chains include duplicate check EP nodes or AP nodes, follow the instructions in "Scaling CC, EP, and DC Nodes without Impacting Non-Scalable Nodes" before you start this procedure.

To scale down the number of Node Manager pods:

1. Open your **override-values.yaml** file for **oc-cn-ocomc**.
2. Specify to wait for the Enhancement Processor (EP) and Distribution Cartridge (DC) nodes to finish processing all input network account records (NARs) before shutting down cartridges. To do so, set the following keys under **ocomcCore.ocomc.configEnv**:
   - **forceWaitForNARSToBeProcessed**: Set this to **true**.
   - **waitForNarsToProcessInSecs**: Set this to a value between 1 and 60. This is the amount of time, in seconds, to wait for NARs to reach the input of a cartridge and then be processed.
3. Specify the number of Node Manager pods to run by doing one of the following:
   - If your Node Manager pods use different data PVs, set the **ocomcCore.ocomc.deployment.nodemgr.count** key to the total number of pods to be up and running. For example, if the count was previously 4, you can scale down the number of pods to 3 or less.
   - If your Node Manager pods use the same data PV, configure the following keys under **ocomcCore.ocomc.deployment.nodemgr**:
     - **ccNMRangeEnd**: The ending range for creating CC Node Manager pods. (The starting range is always **1**).
     - **epdcNMRangeStart**: The starting range for the EP and DC Node Manager pods.
     - **epdcNMRangeEnd**: The ending range for the EP and DC Node Manager pods. This range must include both scalable and non-scalable EP Node Manager pods. Also, the non-scalable EP Node Manager must be the first or last Node Manager pod in the EPDC range.

     For example, if **ccNMRangeEnd** is **2**, **epdcNMRangeStart** is **100**, and **epdcNMRangeEnd** is **100**, the following Node Manager pods would be created:

  - node-mgr-app (CC Node Manager)
  - node-mgr-app-2 (CC Node Manager)
  - node-mgr-app-100 (EP and DC Node Manager)

> **Note:**
>
> The ranges for the CC Node Manager pods and the EP and DC Node Manager pods should not overlap.

4. Specify to run the Node Manager scale down and post scale down jobs. To do so, set the following keys under **ocomc.job**:
   - **scaleDownNM.flag**: Set this to **true**.
   - **scaleDownNM.startAllNodes**: Set this to **yes** if all cartridges in the Administration Server need to be started. Otherwise, set this to **no**.
   - **postScalingDownNM.flag**: Set this to **true**.
   - **postScalingDownNM.backupNeeded**: Specify whether to create a backup of your Offline Mediation Controller installation before scaling down the Node Manager pods. Possible values are **yes** or **no**.
5. Close your **override-values.yaml** file.
6. Run the **helm upgrade** command to update your Offline Mediation Controller release:

   ```
   helm upgrade ReleaseName oc-cn-ocomc --namespace NameSpace --timeout
   duration --values OverrideValuesFile
   ```

   where:
   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the namespace in which to create Offline Mediation Controller Kubernetes objects. To integrate the Offline Mediation Controller cloud native deployment with the ECE and BRM cloud native deployments, they must be set up in the same namespace.
   - *duration* is the amount of time Kubernetes waits for a command to complete, such as **15m** for 15 minutes or **10m** for 10 minutes. The default is 5 minutes.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.
7. In your **override-values.yaml** file for **oc-cn-ocomc**, set the following keys:
   - **ocomcCore.ocomc.job.scaleDownNM**: Set this to **false**.
   - **ocomcCore.ocomc.job.postScalingDown**: Set this to **false**.
   - **ocomcCore.ocomc.configEnv.forceWaitForNARSToBeProcessed**: Set this to **false**.

Scaling CC, EP, and DC Nodes without Impacting Non-Scalable Nodes

> **Note:**
>
> This functionality is supported in Offline Mediation Controller 15.0.

If your Node Manager pods share the same data PV, you can scale up or scale down the number of Collection Cartridge (CC), Enhancement Processor (EP), and Distribution Cartridge (DC) nodes without impacting the following non-scalable Offline Mediation Controller nodes: Duplicate Check EP nodes and Aggregation Processor (AP) nodes.

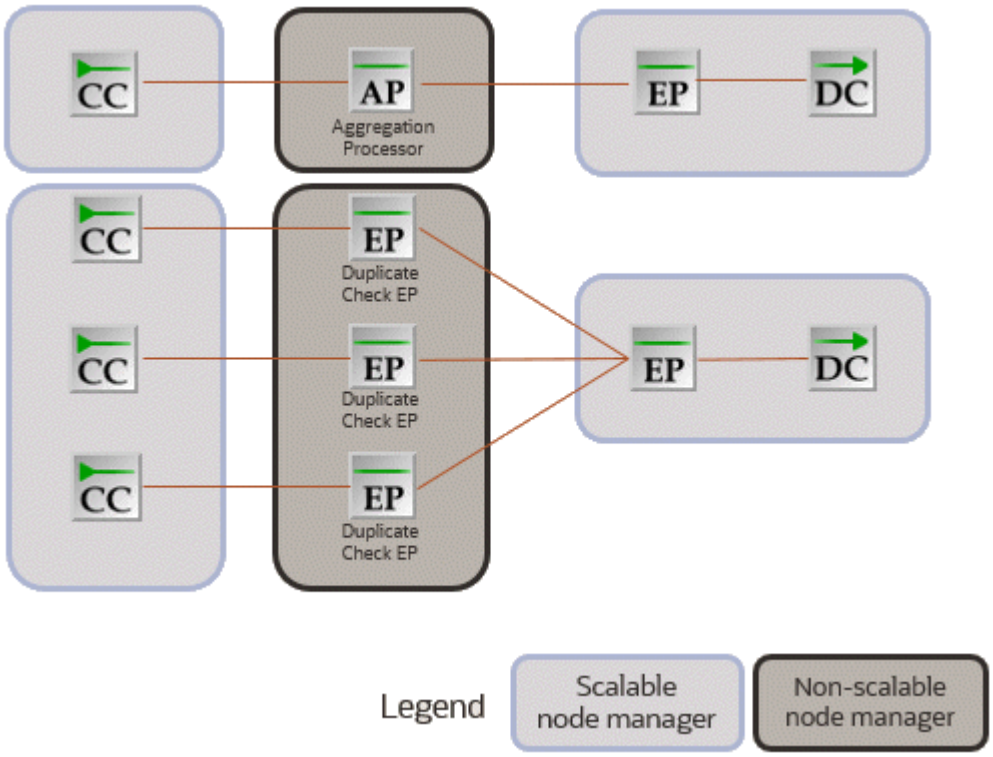
To do so, you must identify separate Node Managers for each of the following:

- Only CC nodes
- Only DC nodes and scalable EP nodes
- Only Duplicate Check EP nodes and AP nodes (This Node Manager will not be replicated)

You must also create separate mediation hosts for each Node Manager, create routes between the CC Node Managers and Duplicate Check EP Node Managers, and create routes between the Duplicate Check EP Node Managers and EP and DC Node Managers. You create mediation hosts and routes by using Administration Client or NMShell. See "Connecting Your Administration Client" and "Using NMShell to Automate Deployment of Node Chains".

Figure 11-1 shows a sample Offline Mediation Controller architecture that contains six Node Managers: two CC Node Managers, one AP Node Manager, one EP Duplicate Check Node Manager, and two EPDC Node Managers.

Figure 11-1 Example Node Manager Architecture with Non-Scalable Nodes



To scale the Node Manager pods without impacting the non-scalable Node Manager pods, follow the instructions in "Scaling Up Node Manager Pods" and "Scaling Down Node Manager Pods", except do the following:

- When specifying the range of EP and DC Node Manager pods to create, include all EP and DC Node Managers and all non-scalable Node Managers.

  For the example in Figure 11-1, you would configure three EPDC Node Manager pods as follows:

  ```
  ocomc:
     deployment:
        nodemgr:
           epdcNMRangeStart: 100
           epdcNMRangeEnd: 102
  ```

  In this case, the following Node Manager pods would be created: node-mgr-app-100 (Non-Scalable Node Manager), node-mgr-app-101 (EP and DC Node Manager), and node-mgr-app-102 (EP and DC Node Manager).

- When configuring a scale up job, set the **EPDC_NM** key to a scalable Node Manager pod.

  For the example in Figure 11-1, you could specify to replicate either node-mgr-app-101 or node-mgr-app-102:

  ```
  ocomc:
     job:
        scaleUpNMSameDataPV:
           EPDC_NM: node-mgr-app-101@node-mgr-app-101:55109
  ```

12

Deploying into Oracle Cloud Infrastructure

Learn how to deploy Oracle Communications Offline Mediation Controller cloud native services into Oracle Cloud Infrastructure.

Topics in this document:

Deploying into Oracle Cloud Infrastructure

Oracle Cloud Infrastructure is a set of complementary cloud services that enable you to run a wide range of applications and services in a highly available hosted environment. It offers high-performance compute capabilities (as physical hardware instances) and storage capacity in a flexible overlay virtual network that is securely accessible from your on-premises network. Among many of its services, the Offline Mediation Controller cloud native deployment is tested in an Oracle Cloud Infrastructure environment using its database and container engine for Kubernetes services on a bare metal instance.

Deploying the Offline Mediation Controller cloud native services into Oracle Cloud Infrastructure involves these high-level steps:

> **Note:**
>
> These are the bare minimum tasks for deploying Offline Mediation Controller cloud native services in Oracle Cloud Infrastructure. Your steps may vary from the ones listed below.

1. Sign up for Oracle Cloud Infrastructure.
2. Create a Kubernetes cluster and deselect the **Tiller (Helm) Enabled** option. The version of Helm used by Oracle Cloud Infrastructure isn't compatible with the Offline Mediation Controller cloud native software requirements.
3. Install and configure the Oracle Cloud Infrastructure Command Line Interface (CLI).

   CLI is a small footprint tool that you can use on its own or with the Console to complete OCI tasks. It's needed here to download the **kubeconfig** file.
4. Install and configure **kubectl** on your system to perform operations on your cluster in Oracle Cloud Infrastructure.
5. The **kubeconfig** file (by default named **config** and stored in the **$HOME/.kube** directory) provides the necessary details to access the cluster using **kubectl** and the Kubernetes Dashboard.

   Download **kubeconfig** to access your cluster on Oracle Cloud Infrastructure by entering this command:

   ```
   oci ce cluster create-kubeconfig --cluster-id ClusterId --file $HOME/.kube/
   config --region RegionId
   ```

where *ClusterId* is the Oracle Cloud Identifier (OCID) of the cluster, and *RegionId* is the region identifier such as us-phoenix-1 and us-ashburn-1.

6. Set the **$KUBECONFIG** environment variable to the downloaded **kubeconfig** file by entering this command:

   ```
   export KUBECONFIG=$HOME/.kube/config
   ```

7. Verify access to your cluster. You can enter this command and then match the output Internal IP Addresses and External IP Addresses against the nodes in your cluster in the Oracle Cloud Infrastructure Console.

   ```
   kubectl get node -o wide
   ```

8. Download and configure Helm in your local system. To install Tiller on your cluster in Oracle Cloud Infrastructure, enter this command:

   ```
   helm init
   ```

9. If you are using a password-protected registry for Docker images, Kubernetes can't pull the images unless the authentication details are provided.

   There are many ways to enable Kubernetes to pull images from a password-protected Docker registry. For example, you could do this on each worker node:

   a. Log in to the Docker registry by entering this command:

      ```
      docker login -u UserName RepoHost:RepoPort
      ```

   b. Copy the **config.json** file where Docker has stored the authentication details to **/var/lib/kubelet**.

10. Place the Offline Mediation Controller cloud native Helm chart on your system where you have downloaded and configured **kubectl** and Helm. Then, follow the instructions in "Installing the Offline Mediation Controller Cloud Native Deployment Package".

13

Building Your Own Images

You can build your own images of Oracle Communications Offline Mediation Controller using the guidance provided in this chapter.

The Docker build commands in this chapter reference Dockerfile and related scripts as is from the **oc-cn-ocomc-docker-files-15.0.0.*x*.0.tgz** package. Ensure that you use your own version of Dockerfile and related scripts before running the build command.

Topics in this document:

- Building Offline Mediation Controller Images

Sample Dockerfiles included in the Offline Mediation Controller cloud native deployment package (**oc-cn-ocomc-docker-files-15.0.0.*x*.0.tgz**) are examples that depict how default images are built for Offline Mediation Controller. If you want to build your own images, refer to the sample Dockerfiles shipped with the product as a reference. Create your own Dockerfiles and then build your images.

> **Caution:**
>
> The Dockerfiles and related scripts are provided for reference only. You can refer to them to build or extend your own Docker images. Support is restricted to core product issues only and no support will be provided for custom Dockerfiles and scripts.

Building Offline Mediation Controller Images

To build images for Offline Mediation Controller, unpack **oc-cn-ocomc-docker-files-15.0.0.*x*.0.tgz** to create the directory structure in **docker_files/**.

Building your own Offline Mediation Controller images involves these high-level steps:

1. You build the Offline Mediation Controller base image. See "Building the Offline Mediation Controller Base Image".
2. You build custom images for Offline Mediation Controller. See "Building Your Offline Mediation Controller Image".

Building the Offline Mediation Controller Base Image

All images from the Offline Mediation Controller cloud native deployment package use Oracle Linux, JDK 1.8, and a few utilities as the base image. Oracle Linux is available from Oracle Container Registry (http://container-registry.oracle.com). You can pull the image from it. To build the base Offline Mediation Controller image, do this:

1. Extract the Docker file package (**oc-cn-ocomc-docker-files-15.0.0.*x*.0.tgz**).

   ```
   tar xvzf oc-cn-ocomc-docker-files-15.0.0.x.0.tgz
   ```

2. Place the **jdk*.tar.gz** in the **docker_files/jdk/** directory.

3. Build the Offline Mediation Controller base image by entering this command from the **docker_files/jdk/** directory:

   docker build -t oc-cn-oraclelinuxjdk:15.0.0.*x*.0 -f Dockerfile.jdk --build-arg PROXY=*ProxyHost*:*Port* .

   For example:

   ```
   docker build -t oc-cn-oraclelinuxjdk:15.0.0.x.0 -f Dockerfile.jdk --build-arg
   PROXY=http://www-proxy.example.com:80 .
   ```

Building Your Offline Mediation Controller Image

To build your Offline Mediation Controller image:

1. Update the Offline Mediation Controller base image (**oc-cn-oraclelinuxjdk:15.0.0.*x*.0**) in the **docker_files/OCOMC/Dockerfile** directory.
2. Move the Offline Mediation Controller 15.0.0.*x*.0 package (**OCOMC-15.0.0.*x*.0_generic_full.jar**) to the **docker_files/OCOMC/container-scripts** directory.
3. Build the Offline Mediation Controller image by entering this command from the **docker_files/OCOMC/** directory:

   docker build -t *Image*:*Tag* **-f Dockerfile .**

   For example:

   ```
   docker build -t oc-cn-ocomc:15.0.0.x.0 -f Dockerfile .
   ```

4. Tag and push the image to your private registry server, if required.