Oracle® Communications Offline Mediation Controller

Cloud Native Installation and Administration Guide

Release 12.0
F29039-07
September 2023

Oracle Communications Offline Mediation Controller Cloud Native Installation and Administration Guide, Release 12.0

F29039-07

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

Diversity and Inclusion

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
- Offline Mediation Controller Cloud Native Deployment Architecture

About the Offline Mediation Controller Cloud Native Deployment

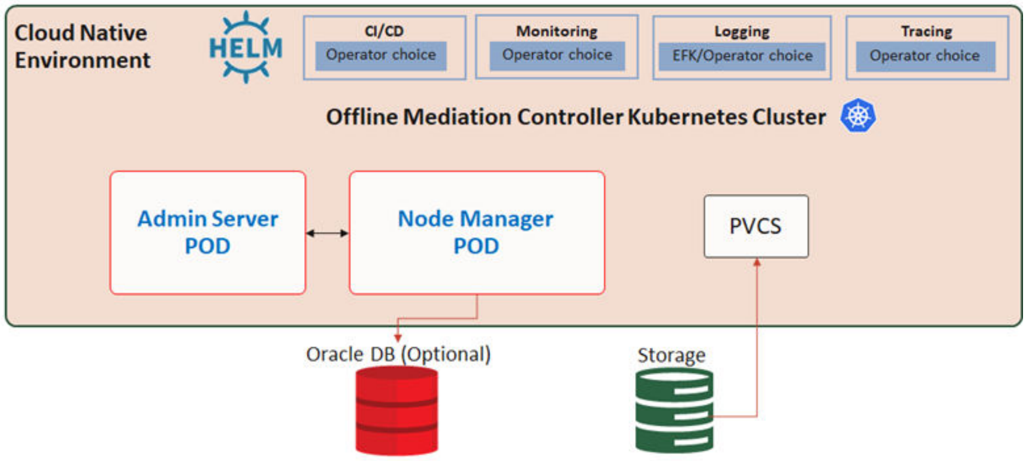
Oracle Communications Offline Mediation Controller now supports its deployment on a cloud native environment. This allows you to harness the benefits of cloud with the services of Offline Mediation Controller. For more information about Offline Mediation Controller, see "Overview of Offline Mediation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

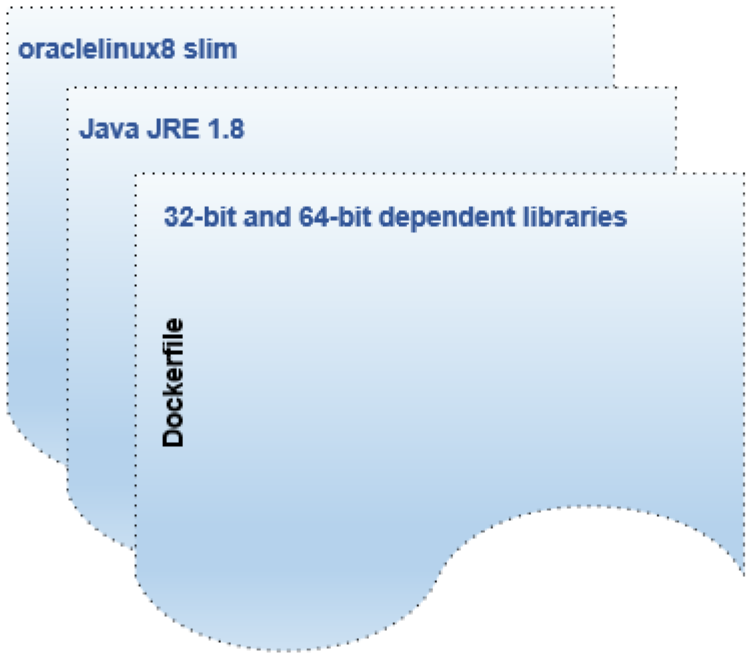


Images and Containers

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
- Software Compatibility
- Environment Setup
- Moving from On-Premise Offline Mediation Controller to Offline Mediation Controller Cloud Native Deployment
- Downloading the Offline Mediation Controller Cloud Native Deployment Package

Overview

The Offline Mediation Controller cloud native deployment package includes ready-to-use images and Helm charts to help you orchestrate containers in Kubernetes. The package also includes sample Dockerfiles and scripts if you want to build the images in your environment.

You can use the Docker images and Helm chart 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 are preconfigured in the Helm charts.

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 12.0.0.*x*.0 with the patch set version number, such as 12.0.0.6.0.

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:12.0.0.*x*.0**	55105
node-mgr-app-5577bbb854-76rx4	Single	**oc-cn-ocomc:12.0.0.*x*.0**	55109

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

Table 2-2 (Cont.) Offline Mediation Controller Services

Service	Service Type	Port	Notes
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

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".

Software Compatibility

For the full list of software that is 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*.
- Offline Mediation Controller, see "Offline Mediation Controller Software Compatibility" in *Offline Mediation Controller Compatibility Matrix*.

Environment Setup

The Offline Mediation Controller cloud native deployment package works with these technologies:

- **Docker:** The Docker platform is used to containerize Offline Mediation Controller products. Docker is needed if you want to build images by writing your own Dockerfiles using the sample Dockerfiles from the Offline Mediation Controller cloud native deployment package. You use the Docker container runtime to create and run containers of these images in Kubernetes. You also host a repository for storing the images, which can later be accessed by Kubernetes for creating pods.

  For more information about Docker, see *Reference Docker* (https://docs.docker.com/reference/).
- **Kubernetes:** Typically, a production-grade Kubernetes cluster is formed by multiple systems distributing the resources. Being a container platform, it provides a container-centric management environment. The cloud native deployment package contains YAML descriptors for creating Kubernetes objects and for allowing it to orchestrate containers of images from Offline Mediation Controller

products. Set up the Kubernetes cluster and secure access to the cluster and its objects with the help of service accounts and proper authentication and authorization modules. Also, choose volumes, a networking model, and logging services to be used in your cluster.

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

> **Note:**
>
> Ensure that your cluster is secured according to standard DevOps practices.

  – **Volumes:** A container’s file system lives only as long as the container does. When a container terminates and restarts, file system changes are lost as well. You shouldn't access the container file system or pods frequently, and it's not easy to share data between container and host systems. 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.

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

  – **Networking:** Kubernetes assumes that pods can communicate with other pods, regardless of which host they land on. Every pod gets its own IP address, so we don't need to explicitly create a link between pods. We almost never need to deal with mapping container ports to host ports. While Kubernetes itself doesn't offer a solution to support its assumption, several implementations are available that meet the fundamental requirements of Kubernetes’ networking model. Choose the networking element depending on the cluster requirement.

- **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. To allow Fluentd to parse your log files, all applications should redirect their logs to STDOUT.

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

- **Helm:** Helm is the package manager for Kubernetes, and chart is a Helm package. Charts, which are packaged as part of the Offline Mediation Controller cloud native deployment package, help create Kubernetes objects like ConfigMap, Secrets, controller sets, and pods with a single command. This feature of Helm makes it easy to handle all Offline Mediation Controller configurations, deployments, and services.

  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.

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

Moving from On-Premise Offline Mediation Controller to Offline Mediation Controller Cloud Native Deployment

If you're an existing Offline Mediation Controller customer on release 6.0 or release 12.0, you can move to the Offline Mediation Controller cloud native deployment. To do so, layer any customizations you made to Offline Mediation Controller on top of the Docker images provided with this release before deploying the images.

Downloading the Offline Mediation Controller Cloud Native Deployment Package

The Offline Mediation Controller cloud native deployment package contains ready-to-use Docker images, a Helm package for deployment in Kubernetes, as well as sample Dockerfiles and scripts that you can use as a reference for building your own images. If you want to build your own images, you must also download the Oracle Communications Offline Mediation Controller installer (**OCOMC-12.0.0.*x*.0_generic_full.jar**).

Note:

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*.

Table 2-3 lists all of the component packages that are included with the Offline Mediation Controller cloud native deployment package.

Table 2-3 Offline Mediation Controller Component Packages

Component Package Name	File Name
Oracle Communications Offline Mediation Controller Image	**oc-cn-ocomc-12.0.0.*x*.0.tar**
Oracle Communications Cloud Native Docker Build Files	**oc-cn-ocomc-docker-files-12.0.0.*x*.0.tgz**
Oracle Communications Cloud Native Offline Mediation Controller Helm Chart	**oc-cn-ocomc-helm-chart-12.0.0.*x*.0.tgz**

3

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-12.0.0.*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.

Extracting the Helm Chart

Before you install Offline Mediation Controller cloud native services, extract the Oracle Communications Cloud Native Helm chart. To do so, enter this command:

```
tar xvzf oc-cn-ocomc-helm-chart-12.0.0.x.0.tgz
```

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 the fully qualified domain name (FQDN) of your private Docker registry (*repoHost*:*repoPort*).
   - *UserName* and *Password* are the user name and password for your private Docker registry.
   - *NameSpace* is the name space 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-helm-chart**:

   ```
   imagePullSecrets: SecretName
   ```

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

   ```
   imageRepository: "RegistryServer"
   ```

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

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 3-1 lists the PersistentVolumeClaims (PVCs) used by the Offline Mediation Controller server.

Table 3-1 List of PVCs in Offline Mediation Controller Server

PVC Name	Default Pod Internal File System
admin-server-pvc	*OMC_home***/config/adminserver**
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 Helm chart (**oc-cn-ocomc-helm-chart**) configures and deploys all of your product services. YAML descriptors in the **oc-cn-ocomc-helm-chart/templates** directory use the **oc-cn-ocomc-helm-chart/values.yaml** file for most of the values. You can override the values by creating an **override-values.yaml** file.

Table 3-2 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 3-2 Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
fromRelease	**upgrade**	The latest base version of Offline Mediation Controller that you have installed on your system in the format **12.0.0.*x*.0**. For example, 12.0.0.4.0 for Patch Set 4, or 12.0.0.5.0 for Patch Set 5. When upgrading from one release to another, this value must be populated. When not performing an upgrade, leave this value empty.
imagePullSecrets	**-**	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 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description

ece.*	**-**	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 to not 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: "12.0.0.x.0"`
		`        imagePullPolicy: IfNotPresent`
		`        clusterName: BRM`
		`        serviceName: ece-server`
		`        persistenceEnabled: "true"`
		`        coherenceClusterPort: ""`

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

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
uniPass	**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	**ocomc.secretEnv**	The password for the Node Manager domain SSL identity key.
nmKeystorepass	**ocomc.secretEnv**	The Offline Mediation Controller Secrets required for SSL and installation.
adminKeypass	**ocomc.secretEnv**	The password for the Administration Server domain SSL identity key.
adminKeystorepass	**ocomc.secretEnv**	The password for the Administration Server domain SSL identity store.
walletPassword	**ocomc.secretEnv**	The string password for opening the wallet.
ocomcPassword	**ocomc.secretEnv**	The Offline Mediation Controller password.
adminServerPassword	**ocomc.secretEnv**	The Administration Server password.
restart_count	**ocomc.configEnv**	Increment the existing value by 1 to re-create pods using the **helm upgrade** command.
name	**ocomc.configEnv**	The name of your Offline Mediation Controller ConfigMap, such as **ocomc-configmap-env**.
sslEnabled	**ocomc.configEnv**	Whether SSL is enabled in your Offline Mediation Controller cloud native environment: **true** or **false**.
eceEnabled	**ocomc.configEnv**	Whether ECE is deployed and enabled in your cloud native environment: **true** or **false**.
ecePath	**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	**ocomc.configEnv**	The location of the Oracle wallet, which contains your TLS certificates. For example: **/home/ocomcuser/ext/**.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
thirdPartyFolder	**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*.
cartridgeFolder	**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	**ocomc.configEnv**	The Node Manager debug mode. The default is **false**.
nmPort	**ocomc.configEnv**	The Node Manager port. The default is **55109**.
nmExternalPort	**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	**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 (Support added in Patch Set 5)	**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 (Support added in Patch Set 4)	**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	**ocomc.configEnv**	The path to the Node Manager domain KeyStore files. Set this key to **/home/ocomcuser/keystore/**, unless you are creating custom images.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
nmDname	**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	**ocomc.configEnv**	The SSL alias for Node Manager. The default is **nodeManager**.
nmKeystoreValidity	**ocomc.configEnv**	The number of days the SSL KeyStore certificate for Node Manager will be valid, such as **365** for one year.
adminsvrPort	**ocomc.configEnv**	The port number for the Administration Server.
adminsvrExternalPort	**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	**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	**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	**ocomc.configEnv**	The host name of the Administration Server node selector.
adminsvrNodeSelectorIp	**ocomc.configEnv**	The external IP address of the Administration Server node selector.
adminsvrNodeSelector	**ocomc.configEnv**	The name of the Administration Server node selector.
adminsvrAuthMode	**ocomc.configEnv**	Whether the Administration Server requires authorization. The default is **false**.
adminsvrDebug	**ocomc.configEnv**	Whether the Administration Server debug mode is turned on. The default is **false**.
adminsvrCartridgeFolder	**ocomc.configEnv**	The name of the directory in which the cartridge pack JAR files reside.
adminsvrPatchFolder	**ocomc.configEnv**	The name of the directory in which your **UDCEnvironment** file and schema files reside.
adminsvrRuleFolder	**ocomc.configEnv**	The name of the directory in which your rule file resides.
adminsvrSharedRuleFolder	**ocomc.configEnv**	The name of the shared rule directory.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
adminsvrDshost	**ocomc.configEnv**	The host name of the Administration Server DS. The default is **localhost**.
adminsvrDsport	**ocomc.configEnv**	The Administration Server DS port. The default is **13001**.
adminsvrAuthuser	**ocomc.configEnv**	Whether the Administration Server must authenticate users.
adminsvrLdapurl	**ocomc.configEnv**	The URL and the LDAP listening port of the Oracle Unified Directory system.
adminsvrLdapdomain	**ocomc.configEnv**	The base DN for the LDAP server.
adminsvrGrpinfo	**ocomc.configEnv**	The **AdminServerImpl.properties** parameter used by the Administration Server.
adminsvrAdmindn	**ocomc.configEnv**	The DN for the Administration Client. For example: uid=Admin,ou=People.
adminsvrMemberval	**ocomc.configEnv**	The **AdminServerImpl.properties** parameter used by the Administration Server.
adminsvrTimeout	**ocomc.configEnv**	The session timeout, in minutes, between the Administration Server and Administration Client. The default is **30**.
adminsvrDisasterRecovery	**ocomc.configEnv**	Whether to configure the ECE DC pod for disaster recovery. The default is **false**.
adminsvrHostBased	**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 (Patch Set 4 Only)". • **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 (Support added in Patch Set 5)	**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**.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
waitForNarsToProcessInSecs (Support added in Patch Set 5)	**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	**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	**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	**ocomc.configEnv**	The DN for the Administration Server. The default is **CN=$HOSTNAME,OU=OracleCloud,O=OracleCorporation,L=RedwoodShores,S=California,C=US**.
adminsvrsslAlias	**ocomc.configEnv**	The alias name for the Administration Server.
adminsvrKeystoreValidity	**ocomc.configEnv**	The number of days the SSL KeyStore certificate for the Administration Server will be valid, such as **365** for one year.
adminclientTruststorePath	**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	**ocomc.configEnv**	If you are creating a fresh installation of 12.0 Patch Set 4 or later, set this to the path and file name of the Offline Mediation Controller 12.0.0.*x*.0 software pack that you downloaded. Set this key to **/container-scripts/OCOMC-12.0.0.*x*.0_generic_full.jar**, unless you are creating custom images.
ocomcSoftwareUpgradePath	**ocomc.configEnv**	If you are upgrading from a previous release to 12.0 Patch Set 4 or later, set this to **/container-scripts/OCOMC-12.0.0.*x*.0_generic_full.jar**, where *x* is the patch set version you are upgrading to. Otherwise, leave this key empty.
inventoryFilePath	**ocomc.configEnv**	The inventory file path.
oudRootUserDn	**ocomc.configEnv**	The DN for the Oracle Unified Directory root user. The default is **cn=Directory Manager**.
oudPath	**ocomc.configEnv**	The path to the Oracle Unified Directory.
oudLdapPort	**ocomc.configEnv**	The port number on which the LDAP server is listening. The default is **1389**.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
oudBaseDn	**ocomc.configEnv**	The DN for the Oracle Unified Directory. The default is **dc=ocomcexample.com**.
adminConnectPort	**ocomc.configEnv**	The Administration Server port for the Oracle Unified Directory. The default is **4444**.
hostName	**ocomc.configEnv**	The host name of the server on which Offline Mediation Controller is deployed. The default is **localhost**.
oracleHome	**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	**ocomc.configEnv**	Whether to regenerate the SSL certificate when the pod restarts. The default is **false**.
gcOptions.* (Support added in Patch Set 5)	**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.* (Support added in Patch Set 5)	**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 (Support added in Patch Set 5)	**ocomc.adminSvrOptions**	The JVM GC settings to apply to the Administration Server pod.
memoryOptions (Support added in Patch Set 5)	**ocomc.adminSvrOptions**	The JVM memory settings to apply to the Administration Server pod.
adminserver.type (Support added in Patch Set 5)	**ocomc.service**	The service type for the Administration Server pod: **ClusterIP** or **NodePort**. The default is **ClusterIP**.
nodemgr.type (Support added in Patch Set 5)	**ocomc.service**	The service type for the Node Manager pod: **ClusterIP** or **NodePort**. The default is **ClusterIP**.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
type (Patch Set 4 only)	**ocomc.service**	The service type for the Administration Server and Node Manager pod: **NodePort** or **ClusterIP**. The default is **ClusterIP**.
name	**ocomc.storageClass**	The storage class name for persistent volume claims (PVCs).
preUpgradeToPS4.flag	**ocomc.job**	Before you upgrade to 12.0 Patch Set 4 or later, set this to **true**.
postUpgradeToPS4.*	**ocomc.job**	After you perform an upgrade to 12.0 Patch Set 4 or later, set the following keys: • **flag**: Set this to **true**. • **existingNM**: Set this to the existing Node Manager pod in the format *name*@*host*:*port*, where *name* is the mediation host's name configured in Node Manager, *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.
copyTrustStore	**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.*	**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. For Patch Set 4 only, also set this key: • **waitTime**: The amount of time to wait between scaling up Node Manager pods. For more information, see "Scaling Node Manager Pods".

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
scaleUpNMSameDataPV.*	**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*. For Patch Set 4 only, also set these keys: • **ccNMCount**: The number of CC nodes to be scaled up, apart from the parent node. • **waitTime**: The amount of time to wait between scaling up Node Manager pods. The default value is **1m** (1 minute). For more information, see "Scaling Node Manager Pods".
scaleDownNM.* (Support added in Patch Set 5)	**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**. For more information, see "Scaling Down Node Manager Pods (Patch Set 5 and Later)".

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
postScalingDownNM.* (Support added in Patch Set 5)	**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**. For more information, see "Scaling Down Node Manager Pods (Patch Set 5 and Later)".
runNMShell.* (Support added in Patch Set 5)	**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. For more information, see "Using NMShell to Automate Deployment of Node Chains (Patch Set 5 and Later)".
nodemgr.* For Node Manager pods that use different data PVs	**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. **Note**: In Patch Set 5 and later releases, set this key only when the **sharedDataPV** key is set to **false**. • **sharedDataPV**: Set this to **false**.

Table 3-2 (Cont.) Offline Mediation Controller Server Keys

Key	Path in values.yaml File	Description
nodemgr.* For Node Manager pods that share the same data PV	**ocomc.deployment**	Information about the Node Manager pods to create. Use these keys only for Node Manager pods that share the same data PV. <ul><li>**ccNMRangeEnd**: The ending rage for creating CC Node Manager pods. (The starting range is always 1).</li><li>**epdcNMRangeStart**: The starting range for creating the EP and DC Node Manager pods.</li><li>**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.</li><li>**sharedDataPV**: Set this to **true**.</li></ul>

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 name space.

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

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

   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-helm-chart --namespace NameSpace --
   values OverrideValuesFile
   ```

   where:

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

- *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-helm-chart --namespace ocgbu --
values override-values.yaml
```

4 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-premise 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-premise 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 post-installation tasks on the Administration Client machine. See "Post-Installation 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-helm-chart**.
3. Set the **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:
   - **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 in different machines.
   - **ocomc.configEnv.adminsvrExternalPort**: Set this to the external port for the Administration Server.
   - **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:

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

Post-Installation Tasks for Administration Client

After you install Administration Client, perform the following post-installation 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 external 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 that 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 the following location to the Administration Client's *OMC_home*/**config/GUI** directory.
   - For Patch Set 4 and later: vol-external PVC on the cloud native environment
   - For Patch Set 3 and earlier: vol-keystore PVC on the cloud native environment

Verifying the Administration Client Connection

Start your Administration Client to verify that 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.

5 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 name space in which the Offline Mediation Controller Kubernetes objects reside.

6 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 **ocomc.configEnv.metricsPort** key in your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**.
- 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 **ocomc.configEnv.metricsPortCN** key in your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**.

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-helm-chart**, 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-helm-chart --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 (Patch Set 5 and Later)

> **Note:**
>
> Running NMShell outside of a pod is supported in Offline Mediation Controller 12.0 Patch Set 5 and later releases.

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 post-install 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-helm-chart**.

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 post-install hook, create a Dockerfile that copies your input files to an accessible location. For example:

     ```
     FROM oc-cn-ocomc:12.0.0.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-helm-chart**, set the following keys under **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.
     - **post-install**: Specifies to run this job as post-install 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 post-install hook, enter this command:

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

   where:

   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the name space 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 name space.
   - *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-helm-chart** 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 name space 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-helm-chart --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 name space 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 name space 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 name space is 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 patch set version of Offline Mediation Controller.

The following procedure assumes that you have upgraded Offline Mediation Controller from Patch Set 5 (Revision 1), to Patch Set 6 (Revision 2), and then to Patch Set 7 (Revision 3). To roll back your upgrade from Patch Set 7 to 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-helm-
   chart     12.0.0.5.0      Initial install
   2           Thu May 30 08:32:09 2030   superseded   oc-cn-ocomc-helm-
   chart     12.0.0.6.0      Upgraded successfully
   3           Thu May 30 09:50:00 2030   deployed     oc-cn-ocomc-helm-
   chart     12.0.0.7.0      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-
   helm-chart    12.0.0.5.0     Initial install
   2          Thu May 30 08:32:09 2030   superseded    oc-cn-ocomc-
   helm-chart    12.0.0.6.0     Upgraded successfully
   3          Thu May 30 09:50:00 2030   superseded    oc-cn-ocomc-
   helm-chart    12.0.0.7.0     Upgraded successfully
   4          Thu May 30 11:25:00 2030   deployed      oc-cn-ocomc-
   helm-chart    12.0.0.6.0     Roll back to 2
   ```

7

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 (Patch Set 5 and Later)

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 (Patch Set 5.1 and Later)" 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 post-install hook.

In Patch Set 5 and later releases, 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 **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-helm-chart**, increment the **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-helm-chart --namespace NameSpace
   --values OverrideValuesFile
   ```

where:

- *ReleaseName* is the release name, which is used to track this installation instance.
- *NameSpace* is the name space 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 name space.
- *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 post-install hook and your node chain configuration files include cartridge JARs, do the following:

   a. In your **override-values.yaml** file, set the **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:12.0.0.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-helm-chart**.

4. Specify the number of Node Manager pods to create:

   - If your Node Manager pods use different data PVs, set the **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 post-install hook, do the following:
   a. Create a Dockerfile that is similar to the following:

   ```
   FROM oc-cn-ocomc:12.0.0.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-helm-chart --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-helm-chart --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-helm-chart**, set the **ocomc.job.scaleUpNMDifferentDataPV.flag** or **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 Up Node Manager Pods (Patch Set 4 Only)

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.

In the Patch Set 4 release, you scale up Node Manager pods by performing these high-level tasks:

1. Creating the Node Manager pods when you initially deploy Offline Mediation Controller.

   See "Creating Node Manager Pods During Deployment".

2. Configuring the Node Manager pods by adding node chains and specifying the Node Manager pods and nodes to replicate.

   See "Configuring Your Node Manager Pods".

3. When you are ready to scale up the number of Node Manager pods, running the scale up job.

   See "Running the Scale Up Job".

Creating Node Manager Pods During Deployment

During the Offline Mediation Controller deployment process, you specify the initial number of Node Manager pods to create and indicate whether the pods will share the same data PersistentVolume (PV).

To create your Node Manager pods:

1. Open your **override-values.yaml** file for **oc-cn-ocomc-helm-chart** in a text editor.
2. Enable your pods to be scaled up by setting the **ocomc.configEnv.adminsvrHostBased** key to **true**.
3. Specify to create one Node Manager pod by setting the **ocomc.deployment.nodemgr.count** key to **1**.

> **Note:**
>
> The count must be **1** for the first deployment. You can increment the number of Node Manager pods later when you run the scale up job.

4. Specify whether the data PV will be shared across Node Manager pods by setting the **ocomc.deployment.nodemgr.sharedDataPV** key to one of the following:
   - **true**: All Node Manager pods will share the same data PV.
   - **false**: The Node Manager pods will have different data PVs.
5. Set any other keys that are needed from Table 3-2.
6. Save and close the file.
7. Run the **helm install** command:

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

   where:
   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the name space 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 name space.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the chart's **values.yaml** file.

The Node Manager pods are up and running in your Offline Mediation Controller cloud native environment.

Configuring Your Node Manager Pods

After Offline Mediation Controller is deployed, configure your Node Manager pods by:

- Creating nodes and node chains. You can create them manually through the Administration Client or by importing them from a node chain configuration file.
- Specifying the Node Manager pods and nodes to replicate.

> **Note:**
>
> The scale up process does not support the replication of Aggregation Processor (AP) and duplicate check Enhancement Processor (EP) nodes. These nodes should not be present in any Node Manager you want to replicate.

To configure your Node Manager pods:

1. If you want to create nodes and node chains by importing them from node chain configuration (**.nmx** and **.xml**) files, do the following:

   > **Note:**
   >
   > - For pods with different data PVs, the configuration files must define a Node Manager that contains all of the nodes.
   > - For pods with the same data PV, the configuration files must define two Node Managers: one containing all of the Collection Cartridge (CC) node instances, and one containing all of the Enhancement Processor (EP) and Distribution Cartridge (DC) node instances with the connected routes between the CC and EP/DC nodes.

   a. Move your configuration files to the vol-external PVC, which has a default path of **/home/ocomcuser/ext**.

   b. Set the permissions for your configuration files to:

   ```
   chown 1000:1000
   chmod 777
   ```

   c. If you are building your own images, create a Dockerfile that is similar to the following and then build the Offline Mediation Controller image:

   ```
   FROM oc-cn-ocomc:12.0.0.x.0
   RUN mkdir -p /home/ocomcuser/install/image
   COPY exportFile.xml /home/ocomcuser/install/image
   COPY exportFile.nmx /home/ocomcuser/install/image
   ```

2. Open your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**.

3. If your Node Manager pods will 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 the Node Manager, *Mediation_host* is the 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.
   - **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**.
   - **scaleUpNMDifferentDataPV.waitTime**: Set this to the amount of time to wait between scaling up Node Manager pods. The default is one minute (**1m**).

4. If your Node Manager pods will 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 the Node Manager, *CCNM_host* is the hostname 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 hostname 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.
   - **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, 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*. This key is mandatory when the **exportConfigFile** key is set.
   - **scaleUpNMSameDataPV.waitTime**: Set this to the amount of time to wait between scaling up Node Manager pods. The default is one minute (**1m**).
5. Save and close the file.
6. If you have multiple CC Node Manager pods sharing the same data PV, do one of the following:
   - Modify the **fileLocation** entry in the **export.xml** file (if you are importing the node chain configuration).
   - Modify the **fileLocation** entry in the **general.cfg** file on node-manager-pvc (if you configured the node chains manually using Administration Client).
   - In the Administration Client, right-click the cartridge, click **Edit**, and add a variable to the file location path (if you configured the node chains manually using Administration Client).

   For example, in the **export.xml** file, you would add **${***VARIABLE***}** to the end of the file location path and then set *VARIABLE* to the appropriate value: **INSTANCE**, **NODEID**, or **HOST**.

   ```
   <configVariable name="fileLocation">/home/ocomcuser/data/$
   {VARIABLE}</configVariable>
   ```

Example 7-1 Configuring Node Manager Pods with Different Data PVs

This shows sample **override-values.yaml** entries for configuring your Node Manager pods. In this example, the Node Manager pods use different data PVs and the parent Node Manager pod already has its node chain up and running.

```
ocomc:
  job:
    scaleUpNMDifferentDataPV:
      flag: "true"
      parent_NM: "node-mgr-app@node-mgr-app:55109"
      exportConfigFile: ""
      waitTime: "1m"
    scaleUpNMSameDataPV:
      flag: "false"
```

Example 7-2 Configuring Node Manager Pods that Share a Data PV

This shows sample **override-values.yaml** entries for configuring your Node Manager pods. In this example, the Node Manager pods all share the same data PV, the node chain configuration will be imported from your **export.nmx** and **export.xml** files, and the Node Manager pods have no nodes yet.

```
ocomc:
  job:
    scaleUpNMDifferentDataPV:
      flag: "false"
    scaleUpNMSameDataPV:
      flag: "true"
      CC_NM: "node-mgr-app@node-mgr-app:55109"
      EPDC_NM: "node-mgr-app-2@node-mgr-app-2:55109"
      exportConfigFile: "/home/ocomcuser/tmp/export"
      importNMList: "node-mgr-app@node-mgr-app:55109,node-mgr-app-2@node-mgr-
app-2:55109"
      waitTime: "1m"
```

Running the Scale Up Job

When you want to scale up the number of Node Manager pods in your Offline Mediation Controller cloud native environment to meet current capacity requirements, you run a scale up job.

To run a scale up job:

1. If 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 **ocomc.cofigEnv.cartridgeFolder** key, which can be set to a directory in the external PV.
   b. In your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**, increment the **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-helm-chart --namespace
NameSpace --values OverrideValuesFile
```

where:

- *ReleaseName* is the release name, which is used to track this installation instance.
- *NameSpace* is the name space 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 name space.
- *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. Create additional Node Manager pods by doing the following:

   a. In your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**, increase the number of Node Manager pods to the desired amount in the **ocomc.deployment.nodemgr.count** key. For example:

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

   b. Set the **ocomc.cofigEnv.cartridgeFolder** key to **/home/ocomcuser/cartridgeJars/**.

   c. Place any cartridge JARs in the **/home/ocomcuser/cartridgeJars/** directory.

   d. Run the **helm upgrade** command to update the Helm release:

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

   Wait until all node-mgr-app pods are up and running.

   e. Verify that the number of running node-mgr-app pods matches the value you specified in the **ocomc.deployment.nodemgr.count** key by running the following command:

   ```
   kubectl get pods
   ```

3. If SSL is enabled in Offline Mediation Controller, do the following:

   a. In your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**, set the **ocomc.job.copyTrustStore.flag** to **true**.

b. Run the **helm upgrade** command to update your Helm release:

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

   Your SSL certificate files are copied to the new Node Manager pods.

4. Restart the admin-server-app pod by doing the following:
   a. Set the following keys in your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**:
      - **ocomc.job.copyTrustStore.flag**: Set this to **false**.
      - **ocomc.configEnv.restart_count**: Increment this value by 1.
   b. Run the **helm upgrade** command to update your Helm release:

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

5. If your Node Manager pods use different data PVs, configure your new Node Manager pods by setting the following keys in your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**:
   - **ocomc.job.scaleUpNMDifferentDataPV.flag**: Set this to **true**.
   - **ocomc.job.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 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.
   - **ocomc.job.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**.
   - **ocomc.job.scaleUpNMDifferentDataPV.waitTime**: Set this to the amount of time to wait between scaling up Node Manager pods. The default is one minute (**1m**).

6. If your Node Manager pods use the same data PV, configure your new Node Manager pods by setting the following keys in your **override-values.yaml** file for **oc-cn-ocomc-helm-chart**:
   - **ocomc.job.scaleUpNMSameDataPV.flag**: Set this to **true**.
   - **ocomc.job.scaleUpNMSameDataPV.CC_NM**: Set this to the Node Manager pod 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 hostname 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.
   - **ocomc.job.scaleUpNMSameDataPV.EPDC_NM**: Set this to the Node Manager pod 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 hostname 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.

- **ocomc.job.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, 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**.
- **ocomc.job.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*. This key is mandatory when the **exportConfigFile** key is set.
- **ocomc.job.scaleUpNMSameDataPV.waitTime**: Set this to the amount of time to wait between scaling up Node Manager pods. The default is one minute (**1m**).

7. Ensure that no other scale up job exists. If a job exists, delete it.
8. Run the **helm upgrade** command to update the Helm release:

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

   Wait until the job completes. You can view the job's progress by checking the **ScalingJob.txt** log files in the *OMC_home*/**log/** directory.

Ensure that you set the following keys in your **override-values.yaml** file to **false** so that the jobs are not run again the next time you update the Helm release: **ocomc.job.scaleUpNMSameDataPV.flag** and **ocomc.job.scaleUpNMDifferentDataPV.flag**.

Scaling Down Node Manager Pods (Patch Set 5 and Later)

> **Note:**
>
> Scaling down Node Manager pods is supported in Offline Mediation Controller 12.0 Patch Set 5 and later releases.

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 (Patch Set 5.1 and Later)" 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-helm-chart**.
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 **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 **ocomc.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 **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-helm-chart --namespace
NameSpace --timeout duration --values OverrideValuesFile
```

   where:

   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the name space 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 name space.
   - *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-helm-chart**, set the following keys:
   - **ocomc.job.scaleDownNM**: Set this to **false**.
   - **ocomc.job.postScalingDown**: Set this to **false**.
   - **ocomc.configEnv.forceWaitForNARSToBeProcessed**: Set this to **false**.

Scaling CC, EP, and DC Nodes without Impacting Non-Scalable Nodes (Patch Set 5.1 and Later)

Note:

This functionality is supported in Offline Mediation Controller 12.0 Patch Set 5.1 and later releases.

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.

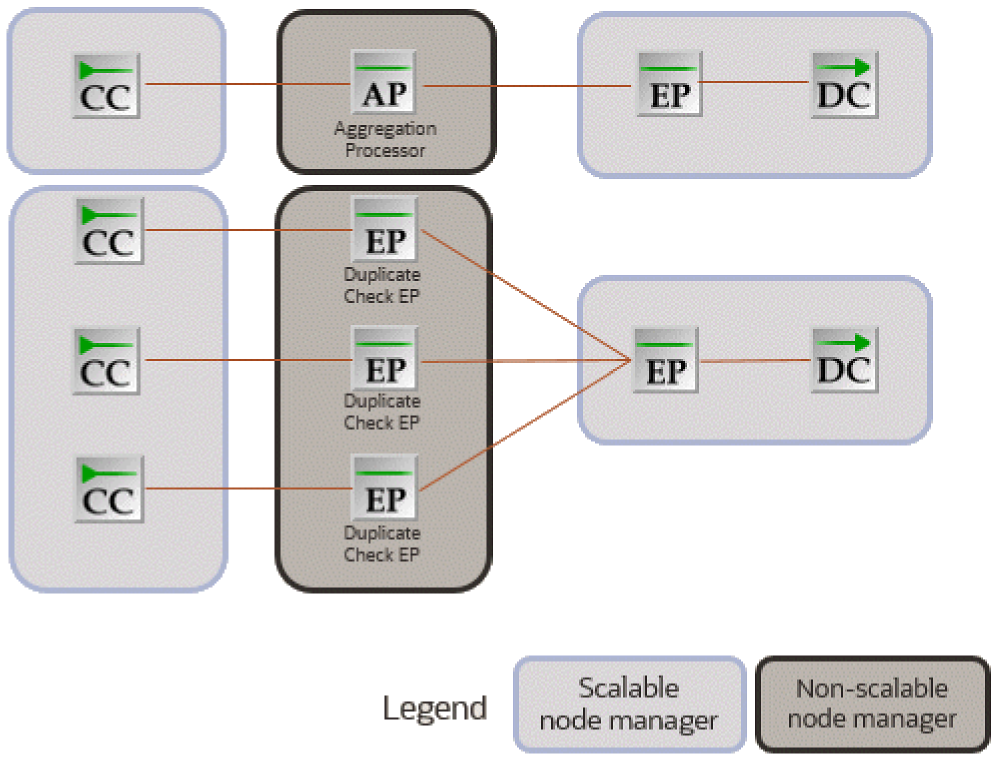
To do so, you must identify separate Node Managers for each of the following:

- 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 (Patch Set 5 and Later)".

Figure 7-1 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 7-1 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 (Patch Set 5 and Later)" and "Scaling Down Node Manager Pods (Patch Set 5 and Later)", 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 7-1, 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 7-1, 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
```

8

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".

9

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-12.0.0.*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-12.0.0.*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-12.0.0.*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 both Docker Hub (https://hub.docker.com/) and Oracle Container Registry (http://container-registry.oracle.com). You can pull the image from either of these repositories. To build the base Offline Mediation Controller image, do this:

1. Extract the Docker file package (**oc-cn-ocomc-docker-files-12.0.0.*x*.0.tgz**).

```
tar xvzf oc-cn-ocomc-docker-files-12.0.0.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:12.0.0.x.0 -f Dockerfile.jdk --build-
   arg PROXY=ProxyHost:Port .
   ```

   For example:

   ```
   docker build -t oc-cn-oraclelinuxjdk:12.0.0.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:12.0.0.*x*.0**) in the **docker_files/OCOMC/Dockerfile** directory.
2. Move the Offline Mediation Controller 12.0.0.*x*.0 package (**OCOMC-12.0.0.*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:12.0.0.x.0 -f Dockerfile .
   ```

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