Oracle® Communications Billing and Revenue Management

Cloud Native Deployment Guide

Release 15.1
F93181-01
April 2025

Oracle Communications Billing and Revenue Management Cloud Native Deployment Guide, Release 15.1

F93181-01

Contents

Preface

This guide describes how to install and administer Oracle Communications Billing and Revenue Management (BRM) Cloud Native Deployment Option.

Audience

This document is intended for DevOps administrators and those involved in installing and maintaining an Oracle Communications Billing and Revenue Management (BRM) Cloud Native Deployment.

Documentation Accessibility

Diversity and Inclusion

Part I
Overview of BRM Cloud Native

This part provides an overview of the Oracle Communications Billing and Revenue Management (BRM) cloud native deployment. It contains the following chapters:

1 Overview of the BRM Cloud Native Deployment

Learn about configuring Oracle Communications Billing and Revenue Management (BRM) to run as a cloud native application in a containerized and orchestrated deployment architecture.

Topics in this document:

- About the BRM Cloud Native Deployment
- BRM Cloud Native Deployment Architecture

About the BRM Cloud Native Deployment

Oracle Communications Billing and Revenue Management (BRM), along with the following BRM applications, are available in a cloud native deployment option, supporting a Kubernetes-orchestrated containerized multi-service architecture to facilitate continuous integration, continuous delivery, and DevOps practices. This allows you to harness the benefits of the cloud with BRM's services.

- Oracle Communications Pricing Design Center (PDC)
- Oracle Communications Elastic Charging Engine (ECE)
- Oracle Communications Pipeline Configuration Center (PCC)
- Oracle Communications Billing Care
- Oracle Communications Business Operations Center

Note:

You can also deploy Oracle Communications Offline Mediation Controller on a cloud native environment. See "About the Offline Mediation Controller Cloud Native Deployment" in *Offline Mediation Controller Cloud Native Installation and Administration Guide* for more information.

You can set up your own BRM cloud native environment or build your own images of BRM and its applications. You use the cloud native deployment package to automate the deployment of BRM products and speed up the process to get services up and running, with product deployments preconfigured to communicate with each other through Helm charts.

BRM Cloud Native Deployment Architecture

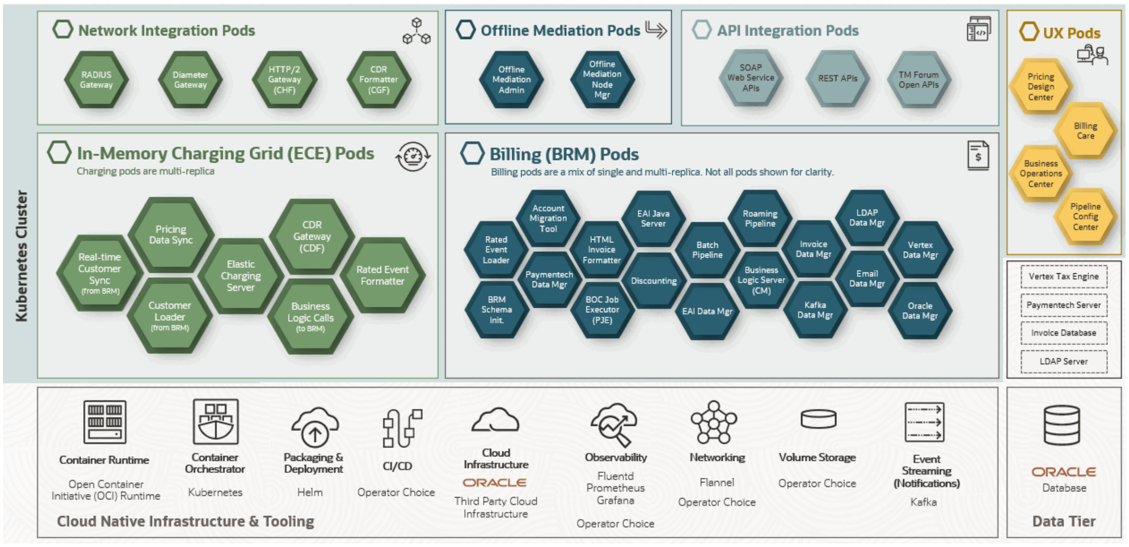
In the BRM cloud native architecture, each BRM service runs as a container and deploys as a Kubernetes pod, which is the fundamental building block of Kubernetes. Many core BRM services can be deployed and managed as multiple replicas within a Kubernetes replica set.

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

Note:

Not all pods are shown for clarity. Pod names are descriptive and may differ from actual names in some cases.

Figure 1-1 BRM Cloud Native Deployment Architecture



In a cloud native deployment:

- PDC, Billing Care, Business Operations Center, and PCC are client applications. They connect to the CM, which represents the business logic layer of BRM, by using the Portal Communications Protocol (PCP).
- The CM communicates with other pods, which represent the data management layer of BRM, by using PCP.
- All PCP communication is encrypted using Transport-Layer Security (TLS).
- The data managers (DMs) interact with other downstream products that run the business logic.

  The downstream products can be containers or an on-premise system.
- ECE rates events and applies charges.
- Rating files for the batch pipeline are fed in through a Kubernetes PersistentVolumeClaim (PVC). The batch pipeline output is also available in a PVC for consumption by the Rated Event (RE) Loader pod.

The figure contains the following pods:

- A Kubernetes cluster, which sits in the middle of the architecture. It includes all Network Integration pods, Offline Mediation pods, API Integration pods, In-Memory Charging Grid (ECE) pods, Billing (BRM) pods, and User Experience (UX) pods.
  - Network Integration pods: This consists of the following:
    - RADIUS Gateway: This is a network-facing RADIUS Server.

    * Diameter Gateway: This is a network-facing Diameter Server.
    * HTTP/2 (CHF) Gateway: This is an integrated Charging Function (CHF) capability that serves 5G standalone (SA) HTTP/2 Service Based Architecture network requests.
    * CDR Formatter (CGF): The call detail record (CDR) formatter extracts unrated 5G SA CDRs from the CDR storage database and formats them for the Kafka messaging service or disk storage. The CHF-CDR format is aligned with 3GPP TS 32.298 v16.5.0.
  – Offline Mediation pods: This consists of the following:
    * Offline Mediation Administrator: This is the Offline Mediation Controller administrator server.
    * Offline Mediation Node Manager: This is the Offline Mediation Controller node manager.
  – API Integration pods: This consists of the following:
    * SOAP Web Service APIs: This supports SOAP Web Services APIs.
    * REST APIs: This supports REST APIs.
    * TM Forum Open APIs: This supports TM Forum-aligned REST APIs.
  – In-Memory Charging Grid (ECE) pods: This consists of the following:
    * Real-time Customer Sync (from BRM): This applies synchronous or real-time customer account data updates from BRM to the in-memory charging grid.
    * Pricing Data Sync: This loads the pricing data from PDC into the charging grid.
    * Customer Loader (from BRM): This supports the initial loading and asynchronous updates of customer, data, credit limit data, offer profiles, product offering cross-reference data, and configuration objects from BRM to the Elastic Charging Engine.
    * Elastic Charging Server: This is the core in-memory grid-based charging service that supports rating, balance management, and session management.
    * CDR Gateway (CDF): This generates 5G SA unrated CDRs by processing requests from the HTTP/2 Gateway and persists them in the CDR storage database.
    * Business Logic Calls (to BRM): This sends information to the BRM server when business logic must be triggered or when updates are made to the BRM database.
    * Rated Event Formatter: This formats the rated events generated from the in-memory charging grid to BRM.
  – Billing (BRM) pods: This consists of the following:
    * Rated Event Loader: This loads the rated events generated by the in-memory charging grid into the BRM Database.
    * BRM Schema Initializer: This initializes the BRM Schema.
    * Account Migration Tool: This moves accounts between BRM schemas.
    * Paymentech Data Manager: This provides an interface to the Paymentech credit card processing service.
    * HTML Invoice Formatter: This is an HTML formatter for invoices.
    * BOC Job Executor (PJE): This processes batch jobs through Business Operations Center.

    * EAI Java Server: This enables you to integrate BRM with other Java-based data-sharing applications in your system.
    * Discounting: This is the real-time discounting for subscription charging and billing.
    * Batch Pipeline: This is a rating engine used for rating large groups of offline charges.
    * EAI Data Manager: This supports the publishing of business events for external Enterprise Application Integration (EAI).
    * Roaming Pipeline: This processes roaming CDRs.
    * Business Logic Server (CM): This is an extensible and configurable business logic server that supports a rich set of revenue management functionality covering billing, invoicing, and accounts receivable processes.
    * Invoice Data Manager: This is a data manager for moving invoices to an external database.
    * Kafka Data Manager: This is a pod that supports integration to a Kafka messaging server for synchronizing internal business notification events to an external system.
    * LDAP Data Manager: This allows you to integrate BRM cloud native with an LDAP directory server. It replicates account and service data in the BRM database to the LDAP database.
    * Email Data Manager: This enables you to send email notifications and invoices to your customers.
    * Vertex Data Manager: This allows you to integrate BRM with the Vertex tax processing software.
    * Oracle Data Manager: This provides the integration between the core BRM business logic server and the Oracle Database. It performs Object Relational Mapping.
  - The User Experience pods: This consists of the following:
    * Pricing Design Center: This is an offer design tool and product catalog that supports charging and billing.
    * Billing Care: This is an agent that assists customer account management and a care application.
    * Business Operations Center: This helps create, schedule, and view the revenue management-related business operations. This includes billing, invoicing, payment collection, refunds, B/L reporting, and custom jobs.
    * Pipeline Configuration Center: This is a web-based application that serves as the user interface for Pipeline Manager.

2

About the BRM Cloud Native Deployment Packages

Learn about the Helm charts and images in the Oracle Communications Billing and Revenue Management (BRM) cloud native deployment package that help you deploy and manage pods of BRM product services in Kubernetes.

Topics in this document:

Overview of the BRM Cloud Native Deployment Package

The BRM cloud native deployment package includes the following:

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

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

About BRM Pods

Table 2-1 lists the pods for BRM whose containers are created and services are exposed through them.

Table 2-1 BRM Pods

Pod Name	Replica Type (1)	Container Port	Container Port Name	Service Type
amt	Single-replica	N/A	N/A	N/A
batch-controller	Multi-replica	N/A	N/A	N/A
batch-wireless-pipe	Single-replica	24001	batchpipe	ClusterIP
brm_apps_jobs	N/A	N/A	N/A	N/A

Table 2-1 (Cont.) BRM Pods

Pod Name	Replica Type (1)	Container Port	Container Port Name	Service Type
brmgateway	Multi-instance, 1 instance per schema	15502	brmgateway	ClusterIP
brm-sdk	Single-replica	N/A	N/A	N/A
cm	Multi-replica	11960 (cm) 11961 (perflib, metrics) 11932 (eai-java-server, metrics)	cm-pcp-port cm-perflib-port eai-prom-port	ClusterIP
config_job	N/A	N/A	N/A	N/A
configloader	Single-replica	N/A	N/A	N/A
customerupdater	Multi-instance, 1 instance per schema	31022	N/A	N/A
diametergateway	Multi-instance, each instance can have 1 replica	3868	N/A	NodePort
dm-eai	Multi-replica	11970	dm-pcp-port	ClusterIP
dm-email	Multi-replica	17777	dm-pcp-port	ClusterIP
dm-fusa	Single-replica	15772	dm-fusa-port	ClusterIP
dm-invoice	Multi-replica	27777	dm-pcp-port	ClusterIP
dm-kafka	Multi-replica	12010 12012 (metrics)	dm-pcp-port dm-prom-port	ClusterIP
dm-ldap	Multi-replica	12850	dm-pcp-port	ClusterIP
dm-oracle	Multi-replica	12950 12951 (perflib, metrics)	dm-pcp-port dm-perflib-port	ClusterIP
dm-prov-telco	Multi-replica	20315	dm-pcp-port	ClusterIP
dm-vertex	Multi-replica	31274	dm-vertex-port	ClusterIP
ecs	Multi-replica	31022	N/A	N/A
emgateway	Multi-replica	15502	N/A	ClusterIP
formatter	Multi-replica	22272	formatter-port	ClusterIP
fusa-simulator	Single-replica	9780 (answer_s, online simulator) 8780 (answer_b, online simulator)	answer-s-port answer-b-port	ClusterIP
httpgateway	Multi-replica	8080	N/A	NodePort
init-db	Single-replica	N/A	N/A	N/A
pje	Multi-replica	31960	pje-pcp-port	ClusterIP

Table 2-1 (Cont.) BRM Pods

Pod Name	Replica Type (1)	Container Port	Container Port Name	Service Type
pricingupdater	Single-replica	9999	N/A	N/A
radiusgateway	Multi-replica	1812	N/A	NodePort
ratedeventformatter	Multi-instance, 1 instance required for each role on each schema (2)	9999	N/A	N/A
realtime-pipe	Multi-replica	24000	rtp	ClusterIP
rel-daemon	Multi-replica	N/A	N/A	N/A
rel-job	Single-replica	N/A	N/A	N/A
rel-manager-job	Single-replica	N/A	N/A	N/A
rated-event-manager	Multi-instance, 1 instance per schema	8080	rem-metrics	N/A
roampipe	Single-replica	24002	roampipe	ClusterIP

Note:

1. The BRM cloud native pods support the following replica types:
   - **Multi-replica**: You can scale these pods at the Kubernetes level to the number of replicas needed.
   - **Single-replica**: You cannot scale these pods.
   - **Multi-instance**: You can scale these pods but not at the Kubernetes level. Instead, a new instance needs to be created and scaling is usually limited.
2. The ratedeventformatter pod requires one instance for each role on each schema. For example, suppose you have two schemas on two sites. In that case, you create primary and secondary instances for each schema in site 1 and primary and secondary instances for each schema in site 2, for a total of 8 instances.

About Client Pods and Images

Table 2-2 lists the pods and images for PDC, PDC REST Services Manager, Pipeline Configuration Center, Billing Care, Business Operations Center, and BRM REST Services Manager.

Note:

For the list of pods and images for Offline Mediation Controller, see "About Offline Mediation Controller Pods and Images" in *Offline Mediation Controller Cloud Native Installation and Administration Guide*.

Table 2-2 Client Pods and Images

Pod	Replica Type [1]	Image [2]	Container Port	Service Type	Access URL
bcws	Multi-replica [4]	**bcws:***tag*	7011 (admin-server) 8001 (managed-serverN and cluster-1) 8080 (monitoring-exporter, if monitoring is enabled)	ClusterIP	*host*:*port*/**bcws**
billingcare	Multi-replica [4]	**billingcare:***tag*	7011 (admin-server) 8001 (managed-serverN and cluster-1) 8080 (monitoring-exporter, if monitoring is enabled)	ClusterIP	*host*:*port*/**bc**
boc	Multi-replica	**boc:***tag*	7011 (admin-server) 8001 (managed-serverN and cluster-1) 8080 (monitoring-exporter, if monitoring is enabled)	ClusterIP	*host*:*port*/**opsdashboard**
brm-rest-services-manager	Multi-replica	**brm-rest-services-manager:***tag*	9090 (HTTP) 8080 (HTTPS) 9060 (adminPort)	ClusterIP	*host*:*port*/**brm**
brm-wsm	Multi-replica	**brm_wsm:***tag*	8080 (HTTP) 8443 (HTTPS)	ClusterIP	*host*:*port*/**metro** *host*:*port*/**configurations/endpoints** *host*:*port*/**configurations/endpoints/default**
pcc	Multi-replica	**pcc:***tag*	7012 (HTTPS)	ClusterIP	*host*:*port*/**pcc**

Table 2-2 (Cont.) Client Pods and Images

Pod	Replica Type [1]	Image [2]	Container Port	Service Type	Access URL
pdc	Single-replica	**pdc:***tag*	8001 (HTTP) [3]	ClusterIP	*host*:*port*/**pdc**
pdcrsm	Multi-replica	**pdcrsm:***tag*	8080	ClusterIP	*host*:*port*/ **productCatalogManagement**
brmdomain	Multi-replica	**brm_wsm_wls:***tag*	7001 (admin-server) 8001 (managed-server*N* and cluster-1) 8080 (monitoring-exporter, metrics)	default default metrics	*host*:*port*/ **BrmWebServices**
webhook	Single-replica	**webhook:***tag*	8080	ClusterIP	N/A

Note:

1. The BRM cloud native pods support the following replica types:
   - **Multi-replica**: You can scale these pods at the Kubernetes level to the number of replicas needed.
   - **Single-replica**: You cannot scale these pods.
   - **Multi-instance**: You can scale these pods but not at the Kubernetes level. Instead, a new instance needs to be created and scaling is usually limited.
2. Replace *tag* with the release version number, such as 15.1.0.0.0.
3. If the PDC user sets the **t3ChannelPort** and **t3sChannelPort** keys in the **values.yaml** file, the HTTP, HTTPS, t3Channel, and t3sChannel ports will be NodePort.
4. The Billing Care and Billing Care REST API pods support multi-replica managed services with scaling done through WebLogic Kubernetes Operator.

About BRM PVCs and Pods

Table 2-3 lists the PVCs and pods in a BRM cloud native deployment.

Table 2-3 List of PVCs in BRM Server

PVC Name	Pods
bcws-domain-domain-pvc	bcws-domain-deployer bcws-domain-admin-server bcws-domain-managed-server*N*
bcws-domain-batch-payment-pvc	bcws-domain-deployer bcws-domain-admin-server bcws-domain-managed-server*N*

Table 2-3 (Cont.) List of PVCs in BRM Server

PVC Name	Pods
billingcare-domain-domain-pvc	billingcare-domain-deployer billingcare-domain-admin-server billingcare-domain-managed-server*N*
billingcare-domain-batch-payment-pvc	billingcare-domain-deployer billingcare-domain-admin-server billingcare-domain-managed-server*N*
boc-domain-domain-pvc	boc-domain-deployer boc-domain-admin-server boc-domain-managed-server*N*
brm-sdk	brm-sdk
cmt-pvc	brm-apps-job pje
common-semaphore	batch-wireless-pipe realtime-pipe roampipe
ctqdir	dm-vertex
custom-job-file	brm-apps-job cm
data	batch-wireless-pipe roampipe
dm-kafka	dm-kafka
fusa-temp	dm-fusa
oms-rel-archive	rel-daemon rel-job
oms-rel-data	rel-daemon rel-manager-job
oms-rel-input	rel-daemon rel-job
oms-rel-reject	rel-daemon rel-job
oms-rem-archive	rated-event-manager rel-manager-job
oms-rem-data	rated-event-manager rel-manager-job
oms-rem-input	rated-event-manager rel-manager-job
oms-rem-reject	rated-event-manager rel-manager-job
oms-uel-archive	batch-controller
oms-uel-input	batch-controller

Table 2-3 (Cont.) List of PVCs in BRM Server

PVC Name	Pods
oms-uel-reject	batch-controller
outputcdr	batch-wireless-pipe rel-daemon rel-job
outputreject	batch-wireless-pipe rel-daemon rel-job
pipelinelog	batch-wireless-pipe roampipe
pdc-app-pvc	PDC pod (PDC Application Container)
pdc-brm-pvc	PDC pod (PDC BRM Integration Pack)
roamoutputcdr	rel-daemon rel-job roampipe
roamoutputreject	rel-daemon rel-job roampipe
service-order	brm-apps-job dm-prov-telco
virtual-time	All pods

Table 2-4 lists the services associated with ECE.

Table 2-4 ECE Services

Service Name	Service Type	Port	Description
ece-brmgateway	ClusterIP	15502	BRM Gateway service
ece-cdrgateway	ClusterIP	8084	CDR Gateway service
ece-dgw	NodePort	3868	Diameter Gateway service
ece-emg	ClusterIP	External port	EM Gateway service
ece-jmx-service-external	NodePort	External port	JMX service
ece-http	NodePort	8080 31500	HTTP Gateway service
ece-monitoringagent-service-external	NodePort	External port	Monitoring agent service
ece-rgw	NodePort	1812	RADIUS Gateway service

Part II
Getting Started with BRM Cloud Native Deployment

This part provides information about getting started with your Oracle Communications Billing and Revenue Management (BRM) cloud native deployment, including installing the prerequisite software and downloading the deployment package. It contains the following chapters:

3

About Configuring and Deploying Your BRM Cloud Native Environment

Learn about the high-level steps for configuring and deploying your Oracle Communications Billing and Revenue Management (BRM) cloud native environment.

Topics in this document:

About Configuring and Deploying BRM Cloud Native

You install the BRM cloud native deployment package by configuring and deploying its Helm charts. The Helm charts include YAML template descriptors for all Kubernetes resources and a **values.yaml** file that provides default configuration values for each chart.

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

Note:

This documentation uses the **override-values.yaml** file name for ease of use, but you can name the file whatever you want.

The BRM cloud native deployment package includes the Helm charts in Table 3-1.

Table 3-1 BRM Cloud Native Helm Charts

Chart Name	Description	When to Use
oc-cn-init-db-helm-chart	This chart initializes and upgrades the database schema for the BRM server. • In initialize mode, it: – Creates users, tablespaces, tables, schemas, views, procedures, indexes, and other database objects needed by BRM Server – Loads seed data • In upgrade mode, it modifies the existing database schema to match the current release's data model.	Use this chart in initialize mode when preparing a new BRM setup and have an empty database schema. Use this chart in upgrade mode when upgrading your schema to the latest release.

Table 3-1 (Cont.) BRM Cloud Native Helm Charts

Chart Name	Description	When to Use
oc-cn-op-job-helm-chart	This chart does the following: • Creates WebLogic Server domains for PDC, Billing Care, the Billing Care REST API, and Business Operations Center. • Installs PDC, Billing Care, the Billing Care REST API, and Business Operations Center in their respective domains. • Populates persistent volumes with domain and application files for sharing between WebLogic Server runtimes. • Creates the following PDC groups: – **PricingDesignAdmin**: This group's users have administrative privileges on PDC. They can perform operations on all PDC UI screens, pricing components, and setup components. – **PricingAnalyst**: This group's users have administrative privileges for pricing components and view-only privileges for setup components. – **PricingReviewer**: This group's users have view-only privileges for all pricing and setup components.	If you want to use Billing Care, Business Operations Center, Pricing Design Center, or the Billing Care REST API, install this chart before you install **oc-cn-helm-chart**.
oc-cn-helm-chart	This chart does the following: • Deploys BRM server, PDC, and PCC. • Starts the WebLogic servers for Billing Care, the Billing Care REST API, and Business Operations Center. • Exposes web clients as services outside of the cluster. • Shares persistent volumes between its services through persistent volume claims.	Install this chart to use the services of the BRM server, PDC, PCC, Billing Care, Business Operations Center, or the Billing Care REST API.
oc-cn-ece-helm-chart	This chart does the following: • Deploys ECE and its services. • Sets up the connection with the BRM server and PDC. • Configures sharing of persistent volumes with the BRM server.	Install this chart to use ECE as your convergent charging solution.

High-Level Installation Tasks

You install BRM cloud native on your system by performing these high-level tasks:

1. Install all prerequisite software for your BRM cloud native environment.

   See "Setting Up Prerequisite Software".

2. Prepare your deployment environment by downloading the BRM cloud native deployment package, extracting the Helm charts, and loading the BRM component images.

   See "Preparing Your BRM Cloud Native Environment".

3. Configure and deploy the BRM database schema in your cloud native environment.

   See "Deploying the BRM Database Schema".

4. Configure the BRM cloud native services that you want to include in your system, including:

   - BRM server and PDC services. See "Configuring BRM Server and PDC Services".
   - BRM and PDC REST services. See "Configuring REST Services".
   - BRM client services such as Billing Care and Business Operations Center. See "Configuring the BRM Client Services".
   - ECE services. See "Configuring ECE Services".

5. Deploy the BRM cloud native services in your cloud native environment.

   See "Deploying BRM Cloud Native Services".

4

Setting Up Prerequisite Software

Learn about prerequisite tasks to perform before installing the Oracle Communications Billing and Revenue Management (BRM) cloud native deployment package, such as installing Podman and Helm.

Topics in this document:

Caution:

Oracle does not provide support for any prerequisite third-party software installation or configuration. The customer must handle any installation or configuration issues related to non-Oracle prerequisite software.

BRM Cloud Native Prerequisite Tasks

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

1. Ensure you have downloaded the latest supported software that is compatible with BRM cloud native.
2. Create a Kubernetes cluster.
3. Install Podman and a container runtime supported by Kubernetes.
4. Install Helm.
5. Create and configure a BRM database.
6. Install and configure an external provisioner.
7. If you plan to deploy Pricing Design Center (PDC), Billing Care, the Billing Care REST API, Business Operations Center, or Web Services Manager deployed on WebLogic:

- Install and configure WebLogic Kubernetes Operator.
- Install an ingress controller.

8. If you plan to deploy Elastic Charging Engine (ECE), install and set up an ingress controller and an egress controller.
9. If you plan to deploy a client UI or REST API, install an Identity Provider (IdP) such as Oracle Identity Cloud Service or Oracle Access Management.
10. If you plan to integrate your BRM cloud native deployment with a Kafka Server, install the Apache Kafka software. See "Apache Kafka Quickstart" on the Apache Kafka website for installation instructions.
11. If you plan to integrate your BRM cloud native deployment with Oracle Analytics Publisher, install Oracle Analytics Publisher. See "Installing the Oracle Analytics Server Software" in *Oracle Analytics Installing and Configuring Oracle Analytics Server* for installation instructions.

> **Note:**
>
> The Oracle Analytics Publisher software was previously named Oracle Business Intelligence (BI) Publisher.

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

The following sections provide more information about the required components and services, the options you can choose from, and how you must set them up for your BRM cloud native environment.

Software Compatibility

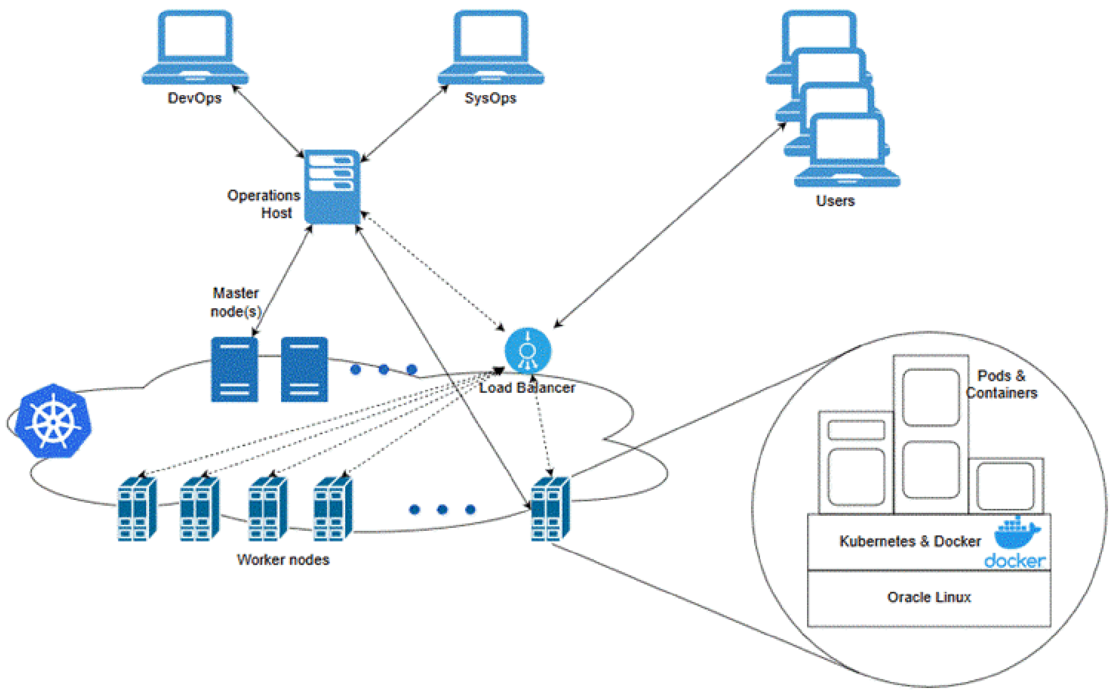
To run, manage, and monitor your BRM cloud native deployment, ensure you use the latest versions of all compatible software. See *BRM Compatibility Matrix*.

Creating a Kubernetes Cluster

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

Figure 4-1 illustrates the Kubernetes cluster and the components that it interacts with.

Figure 4-1 Overview of the Kubernetes Cluster



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

- **Volumes**: Volumes are directories that are accessible to the containers in a pod and provide a way to share data. The BRM cloud native deployment package uses persistent volumes for sharing data in and out of containers, but does not enforce any particular type. You can choose from the volume type options available in Kubernetes.
- **A networking model**: Kubernetes assumes that pods can communicate with other pods, regardless of which host they land on. Every pod gets its own IP address, so you do not need to explicitly create a link between pods or map container ports to host ports. Several implementations are available that meet the fundamental requirements of Kubernetes' networking model. Choose the networking model depending on the cluster requirement.

For more information about Kubernetes, see "Kubernetes Concepts" in the Kubernetes documentation.

Installing Podman

You use the Podman platform to containerize BRM products. Install Podman if you want to do one of these:

- Use the prebuilt images provided with the BRM cloud native deployment package.
- Build your own BRM images by writing your own Dockerfiles using the sample Dockerfiles from the BRM cloud native deployment package.

You can use Podman or any container runtime that supports the Open Container Initiative if it supports the Kubernetes version specified in *BRM Compatibility Matrix*.

Installing Helm

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

The following shows sample steps for installing and validating Helm:

1. Download the Helm software from https://github.com/helm/helm/releases.

   For the list of supported Helm versions, see *BRM Compatibility Matrix*.

2. Extract the Helm files from the archive:

   ```
   tar -zxvf helm-version-linux-amd64.tar.gz
   ```

   where *version* is the Helm version number.

3. Find the helm binary in the unpacked directory and move it to your desired directory. For example:

   ```
   mv linux-amd64/helm /usr/local/bin/helm
   ```

4. Check the version of Helm:

   ```
   helm version
   ```

Helm leverages **kubeconfig** for users running the **helm** command to access the Kubernetes cluster. By default, this is **$HOME/.kube/config**. Helm inherits the permissions set up for this access into the cluster. If role-based access control (RBAC) is configured, you must grant Helm users sufficient cluster permissions.

For more information about installing Helm, see "Installing Helm" in the Helm documentation.

Creating and Configuring Your BRM Database

You must install an Oracle database accessible through the Kubernetes network so BRM cloud native pods can perform database operations. The Oracle database you use can be:

- On-premises, which can be either physical or VM
- Cloud-based, such as Bare Metal, VM, DBaaS on Oracle Cloud Infrastructure, or Oracle Autonomous Database

You can use an existing BRM database or create a new one. See *BRM Compatibility Matrix* for the latest supported database versions.

To create and configure a new BRM database:

1. When you install the Oracle database software, pay particular attention to the following requirements:
   - Install **Oracle Enterprise Edition**
   - Choose a **Customized** installation. This option lets you configure Oracle with the **AL32UTF8** database character set.
   - To configure discounts in BRM, install the following Oracle components:
     – **Oracle XML DB**. For more information, see the Oracle documentation.

   - **Oracle XML Developer's Kit (XDK)**. For more information, see the Oracle documentation.
- Install Oracle JServer as part of the Oracle Database installation.
- To partition the tables in your BRM database, install the **Oracle Partitioning** component. See "Partitioning Tables" in *BRM System Administrator's Guide*.

2. When you create your BRM database, pay particular attention to the following requirements:
   - Specify a **Global Database Name** using the format *DatabaseName***.***DomainName*, where *DatabaseName* is the database name and *DomainName* is the network domain in which the database is located. For example, **pindb***hostname***.example.com**. Most BRM databases use a *DatabaseName* of **pindb***hostname*, but you can use another name.

     Note:

     You can modify your machine's default domain name in the **$ORACLE_HOME/network/admin/sqlnet.ora** file. For information, see your Oracle documentation.

   - Specify a **System Identifier (SID)** for your database. For clarity, it should be the same as your Oracle database name. Most BRM databases are named **pindb***hostname*, but you can use another name.
     - Set the **Character Set** to **AL32UTF8**.
     - Set the **National Character Set** to **AL16UTF16**.
3. Set your LD_LIBRARY_PATH environment variable to **$ORACLE_HOME/lib**.
4. (Optional) Set up TLS authentication in the BRM database. See "Configuring Transport Layer Security Authentication" in *Oracle Database Security Guide*. Also, ensure that you:
   - Create a TLS certificate or obtain one from a certificate provider
   - Install the certificate in the Oracle Database Server
5. Configure your database manually or let the **oc-cn-init-db-helm-chart** Helm chart configure the database for you.

   Note:

   For production systems, your database administrator must create the database manually.

   You can do one of the following:

   - **Use the oc-cn-init-db-helm-chart Helm chart to configure a demonstration database for you**

     If you provide the system administrator's user name and password, the init-db Helm chart can automatically configure your database for demonstration or development systems. For more information, see "Using the BRM Installer to Configure Your Database for Demonstration Systems" in *BRM Installation Guide*.
   - **Configure a demonstration database manually**

You can configure your database manually so it contains additional or larger tablespaces. For more information, see "Configuring Your Database Manually for Demonstration Systems" in *BRM Installation Guide*.

- **Configure a production database manually**

  For production systems, your database administrator must create the tablespaces for the BRM data and indexes. For information on estimating your database size and creating tablespaces, see "Planning Your Database Configuration" in *BRM Installation Guide*. To set the default storage model values for your cloud native installation, set the values in the **configmap_partition_cfg.yaml** file instead of the **pin_tables.values** file.

6. (For Autonomous Databases only) Enable Java on your Autonomous Database by following the instructions in "Enable Oracle Java" in *Oracle Cloud Using Oracle Autonomous Database Serverless*.
7. Grant the BRM schema user select permission on the V$SESSION database table. To do so, connect to the Oracle database with SQL*Plus as the **system** user and then enter this command:

   ```
   SQL> GRANT SELECT ON TABLE V$SESSION TO brmSchemaUser;
   ```

The installers for PDC, Billing Care, and all other products automatically create the tablespaces and users that are required for those products.

Installing an External Provisioner

An external provisioner creates shared, persistent storage for the containers in your BRM cloud native environment. It stores:

- Input data, such as pricing XML files
- Output data, such as archive files and reject files from Rated Event Loader and Universal Event Loader
- Data that needs to be shared between containers, such as **pin_virtual_time**

Install and set up an external provisioner with ReadWriteMany access in your system that provisions volumes dynamically.

Installing WebLogic Kubernetes Operator

Oracle WebLogic Kubernetes Operator helps you to deploy and manage WebLogic domains in your Kubernetes environment. It consists of several parts:

- The operator runtime
- The model for a Kubernetes customer resource definition (CRD)
- A Helm chart for installing the operator

In the BRM cloud native environment, you use WebLogic Kubernetes Operator to maintain the domains and services for Billing Care, the Billing Care REST API, PDC, and Business Operations Center.

The following shows sample steps for installing WebLogic Kubernetes Operator on your BRM cloud native environment:

1. Add the Helm repository for WebLogic Kubernetes Operator:

   ```
   helm repo add weblogic-operator https://oracle.github.io/weblogic-kubernetes-
   operator/charts
   ```

2. Create a new namespace for WebLogic Kubernetes Operator. For example, this **kubectl** command creates the namespace **operator**:

   ```
   kubectl create namespace operator
   ```

3. Install WebLogic Kubernetes Operator:

   ```
   helm install weblogic-operator weblogic-operator/weblogic-operator --namespace
   operator --version version
   ```

   where *version* is the version of WebLogic Kubernetes Operator, such as **4.2.13**. See *BRM Compatibility Matrix* for a list of supported versions.

   If the installation is successful, you will see something similar to this:

   ```
   NAME: weblogic-operator
   LAST DEPLOYED: Tue Oct 6 08:29:03 2020
   NAMESPACE: weblogic-operator
   STATUS: deployed
   REVISION: 1
   TEST SUITE: None
   ```

4. Check the pod:

   ```
   kubectl get pods
   ```

   You should see something similar to this:

   ```
   NAME                                 READY   STATUS    RESTARTS   AGE
   weblogic-operator-849cc6bdd8-vkx7n   1/1     Running   0          57s
   ```

For more information about WebLogic Kubernetes Operator, see "Introduction" in the WebLogic Kubernetes Operator documentation.

Installing an Ingress Controller

Using an ingress controller exposes BRM services outside the Kubernetes cluster and allows clients to communicate with BRM.

The ingress controller monitors the ingress objects and acts on the configuration embedded in these objects to expose BRM HTTP and T3 services to the external network. Adding an external load balancer provides highly reliable single-point access to the services exposed by the Kubernetes cluster. In this case, the ingress controller exposes the services on behalf of the BRM cloud native instance. Using a load balancer removes the need to expose Kubernetes node IPs to the larger user base, insulates users from changes (in terms of nodes appearing or being decommissioned) to the Kubernetes cluster, and enforces access policies.

If you are using Billing Care, the Billing Care REST API, or Business Operations Center, you must add a load balancer to your BRM cloud native system that has:

- Path-based routing for the WebLogic Cluster service.
- Sticky sessions enabled. That is, if the load balancer redirects a client's login request to Managed Server 1, all subsequent requests from that client are redirected to Managed Server 1.
- TLS enabled between the client and the load balancer to secure communications outside of the Kubernetes cluster.

  Business Operations Center and Billing Care use HTTP and rely on the load balancer to terminate HTTPS.

See "Ingress" in the WebLogic Kubernetes Operator documentation for more information about setting up an ingress controller and sample load balancers.

Setting Up ECE Cloud Native Ingress and Egress Flows

Ingress and egress controllers expose ECE services outside the Kubernetes cluster, allowing external networks to communicate with ECE. For example, an ingress controller can route requests from Diameter and 5G HTTP clients to the httpgateway and diametergateway pods for processing. Likewise, an egress controller can send CDR records from the cdrformatter pod to the ECE database.

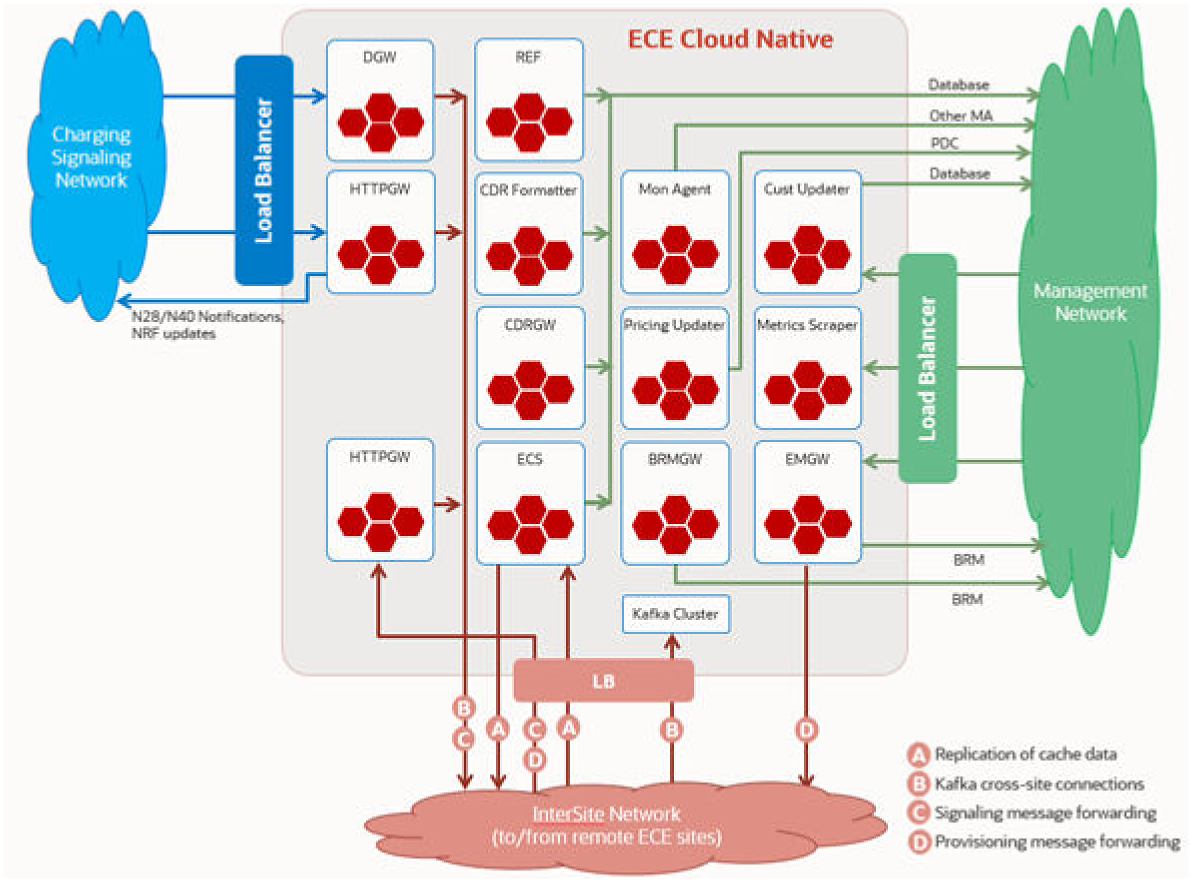
You can expose external network IPs for ingress traffic from external clients using the following:

- A load balancer exposing the IPs on the external network. The load balancer sends ingress traffic to Kubernetes services or node ports.
- Kubernetes service IPs or worker node IPs residing on the external network.

You can route egress traffic through an external network IP that is hosted by a worker node interface.

Figure 4-2 shows an ECE cloud native deployment with sample ingress and egress flows.

Figure 4-2 ECE Cloud Native Ingress and Egress Flows


Charging Signaling Network
Load Balancer
ECE Cloud Native
DGW
REF
HTTPGW
CDR Formatter
Mon Agent
Cust Updater
CDRGW
Pricing Updater
Metrics Scraper
HTTPGW
ECS
BRMGW
EMGW
Kafka Cluster
LB
N28/N40 Notifications, NRF updates
Database
Other MA
PDC
Database
Management Network
Load Balancer
BRM
BRM
InterSite Network (to/from remote ECE sites)
A Replication of cache data
B Kafka cross-site connections
C Signaling message forwarding
D Provisioning message forwarding

In this figure:

- The ingress flows traverse a load balancer, but you can use an alternate ingress flow to meet your business requirements.

- The egress flows are depicted generically. Network source addressing and routing may vary based on your business requirements.
- ECE cloud native uses logical external networks. The number and content of these networks may vary depending on your business requirements.

Table 4-1 describes the egress flow from each ECE pod to an endpoint.

Table 4-1 ECE Cloud Native Egress Flows

ECE Pod	Egress Endpoints
brmgateway	ECE database BRM database **Note:** The BRM database is accessed during installation.
cdrformatter	ECE database **Note:** The ECE database is used for CDR management.
cdrgateway	ECE database **Note:** The ECE database is used for CDR management.
configloader	ECE database
customerupdater	ECE database BRM database
diametergateway	Remote HTTP Gateway (for active-active deployments only) Remote Kafka server (for active-active deployments only) **Note:** This pod does not initiate Diameter connections to Diameter signaling clients.
ece-customerloader-job	ECE database BRM database
ece-persistence-job	ECE database
ece-persistence-upgrade-job	ECE database
ecs	ECE database BRM database Remote ECE Coherence Federation (for active-active and active-standby deployments only) **Note:** The BRM database is accessed during customer loading.
emgateway	BRM database Remote HTTP Gateway (for active-active deployments only) **Note:** This pod forwards requests to the remote HTTP Gateway. This is optional for active-active deployments.
httpgateway	Charging signaling clients Remote HTTP Gateway (for active-active deployments only) Remote Kafka server (for active-active deployments only) **Note:** This pod sends HTTP/2 requests to 5G clients. **Note:** The httpgateway pod's egress to a remote ECE HTTP Gateway is needed only if it is processing 5G charging traffic.
monitoringagent	Monitoring Agents Remote Monitoring Agent (for active-active and active-standby deployments only)
pricingupdater	Pricing Design Center

Table 4-1 (Cont.) ECE Cloud Native Egress Flows

ECE Pod	Egress Endpoints
ratedeventformatter	ECE database BRM database (for Rated Event Manager plug-in only) **Note:** Direct access to the BRM database occurs only when the Rated Event Manager plug-in is configured to write rated events directly to the BRM database.

5

Preparing Your BRM Cloud Native Environment

Learn how to prepare your system for the Oracle Communications Billing and Revenue Management (BRM) cloud native deployment by downloading the BRM cloud native Helm charts and BRM images.

Topics in this document:

Tasks for Preparing Your BRM Cloud Native Environment

Prepare your system for the BRM cloud native deployment by performing the following high-level tasks:

1. Downloading the Helm charts for the BRM cloud native deployment. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Downloading the BRM cloud native images in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. If you plan to deploy Pricing Design Center (PDC), Billing Care, the Billing Care REST API, or Business Operations Center, downloading the Oracle WebLogic cloud native image. See "Pulling WebLogic Images for PDC, Billing Care, Billing Care REST API, and Business Operations Center".

Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files

To download the BRM cloud native Helm charts and Docker files:

1. Go to https://edelivery.oracle.com.
2. Sign in to the Oracle Software Delivery website using an Oracle account.
3. Search for and select the following releases and then click **Continue**.
   - Oracle Communications Billing and Revenue Management Cloud Native Deployment Option 15.1.*x.x.x*

- Oracle Communications Elastic Charging Engine Cloud Native Deployment Option 15.1.*x.x.x*
- Oracle Communications Pricing Design Center Cloud Native Deployment Option 15.1.*x.x.x*

4. Select the following in the download queue and click **Continue**:
   - Oracle Communications Billing and Revenue Management Cloud Native Deployment Option 15.1.*x.x.x*-CN
   - Oracle Communications Elastic Charging Engine Cloud Native Deployment Option 15.1.*x.x.x*-CN
   - Oracle Communications Pricing Design Center Cloud Native Deployment Option 15.1.*x.x.x*-CN
5. Accept the Oracle standard terms and restrictions and then click **Continue**.
6. Select the following packages and then click **Download**:
   - Oracle Communications Cloud Native Helm Chart 15.1.*x.x.x*
   - Oracle Communications Elastic Charging Engine Cloud Native Deployment Option Helm Chart 15.1.*x.x.x*
   - Oracle Communications Cloud Native Operator Job Helm Chart 15.1.*x.x.x*
   - Oracle Communications Cloud Native Database Initializer Helm Chart 15.1.*x.x.x*
   - Oracle Communications Cloud Native Docker Build Files 15.1.*x.x.x*
   - Oracle Communications Elastic Charging Engine Cloud Native Docker Files 15.1.*x.x.x*
   - Oracle Communications Cloud Native Pricing Design Center 15.1.*x.x.x*

   Each package is downloaded to a separate Zip file.
7. Extract the following Helm chart and Docker archive files from each Zip file:
   - BRM Helm Chart: **oc-cn-helm-chart-15.1.*x.x.x*.tgz**
   - ECE Helm Chart: **oc-cn-ece-helm-chart-15.1.*x.x.x*.tgz**
   - Operator Job Helm Chart: **oc-cn-op-job-helm-chart-15.1.*x.x.x*.tgz**
   - Database Initializer Helm Chart: **oc-cn-init-db-helm-chart-15.1.*x.x.x*.tgz**
   - BRM Dockerfiles: **oc-cn-docker-files-15.1.*x.x.x*.tgz**
   - ECE Dockerfiles: **oc-cn-ece-docker-files-15.1.*x.x.x*.tgz**
8. Extract the Helm charts and Dockerfiles from the archive files by running these commands:

```
tar xvzf oc-cn-helm-chart-15.1.x.x.x.tgz

tar xvzf oc-cn-ece-helm-chart-15.1.x.x.x.tgz

tar xvzf oc-cn-op-job-helm-chart-15.1.x.x.x.tgz

tar xvzf oc-cn-init-db-helm-chart-15.1.x.x.x.tgz

tar xvzf oc-cn-docker-files-15.1.x.x.x.tgz

tar xvzf oc-cn-ece-docker-files-15.1.x.x.x.tgz
```

   Table 5-1 lists the files and directories extracted from the archive files.

Table 5-1 Extracted Files

Archive File	Extracted Directories
oc-cn-helm-chart-15.1.*x.x.x*.tgz	**oc-cn-helm-chart** directory: Contains the BRM Helm chart files. **sample_configurations** directory: This directory contains the default configuration XML files, such as **bus_params_AR.xml** and **pin_config_export_gl.xml**.
oc-cn-ece-helm-chart-15.1.*x.x.x*.tgz	**oc-cn-ece-helm-chart** directory: Contains the ECE Helm chart files.
oc-cn-op-job-helm-chart-15.1.*x.x.x*.tgz	**oc-cn-op-job-helm-chart** directory: Contains the WebLogic Operator Job Helm chart files.
oc-cn-init-db-helm-chart-15.1.*x.x.x*.tgz	**oc-cn-init-db-helm-chart** directory: Contains the Database Initializer Helm chart files.
oc-cn-docker-files-15.1.*x.x.x*.tgz	**oc-cn-docker-files** directory: Contains the Dockerfiles for BRM, PDC, PDC REST Services Manager, Pipeline Configuration Center, Business Operations Center, and Billing Care.
oc-cn-ece-docker-files-15.1.*x.x.x*.tgz	**docker_files** directory: Contains the Dockerfiles for ECE.

Pulling BRM Images from the Oracle Container Registry

To pull BRM cloud native images, such as the Connection Manager (CM) image and the Data Manager (DM) image, from the Oracle Container Registry, do the following:

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

   Note:

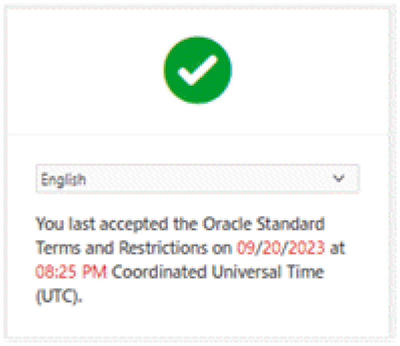
   To pull images for licensed software on the Oracle Container Registry, you must have an Oracle account. You can create an Oracle account at https://profile.oracle.com/myprofile/account/create-account.jspx.

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

   The Oracle Communications Cloud Scale Monetization page appears.
4. Select one of the repository names from Table 5-2.

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

   If successful, you will see something similar to this:



You last accepted the Oracle Standard Terms and Restrictions on 09/20/2023 at 08:25 PM Coordinated Universal Time (UTC).

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

   ```
   podman login container-registry.oracle.com
   ```

7. When prompted for a user name and password, enter your Oracle credentials.
8. Pull the BRM cloud native image from the registry:

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

   where:

   - *imageName* is the name of a software image listed in Table 5-2.
   - *tag* is the tag name for the BRM cloud native image, such as **15.1.***x.x.x*.

   For example, to pull the CM cloud native image from the registry:

   ```
   podman pull container-registry.oracle.com/communications_monetization/cm:15.1.x.x.x
   ```

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

9. Confirm the images have been pulled from the Oracle Container Registry:

   ```
   podman images
   ```

   If successful, you will see something similar to this:

   ```
   REPOSITORY                                                         TAG
   IMAGE ID        CREATED
   container-registry.oracle.com/communications_monetization/cm         15.1.x.x.x
   133dd3580b87     2 seconds ago
   container-registry.oracle.com/communications_monetization/dm_kafka   15.1.x.x.x
   136dd3593b47     3 seconds ago
   ```

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

    ```
    podman logout container-registry.oracle.com
    ```

Table 5-2 lists the image names for the BRM cloud native components.

Table 5-2 BRM Cloud Native Images

Component Name	Image Name
Batch Controller	**batch_controller**
Batch Pipeline	**batch_pipeline**

Table 5-2 (Cont.) BRM Cloud Native Images

Component Name	Image Name
Billing Care	**billingcare**
Billing Care REST API	**bcws**
BRM Applications	**brm_apps**
BRM REST Services Manager	**brm-rest-services-manager**
BRM SDK	**brm_sdk**
Business Operations Center	**boc**
Connection Manager	**cm**
Database Initializer	**init_db**
Database Upgrade	**upgrade**
Elastic Charging Engine	**oc-cn-ece**
Email Data Manager	**dm_email**
Enterprise Application Integration Data Manager	**dm_eai**
Enterprise Application Integration Java Server	**eai_js**
Invoice Data Manager	**dm_invoice**
Invoice Formatter	**formatter**
Kafka Data Manager	**dm_kafka**
LDAP Data Manager	**dm_ldap**
Oracle Data Manager	**dm_oracle**
Paymentech Data Manager	**dm_fusa**
Paymentech Simulator	**answer**
Performance Libraries	**perflib**
Pipeline Configuration Center	**pcc**
Pricing Design Center	**pdc**
Pricing Design Center REST Services Manager	**pdcrsm**
Provisioning Data Manager	**dm_prov_telco**
Rated Event Loader	**rel_daemon**
Rated Event Manager	**rem**
Real-Time Pipeline	**realtimepipe**
Roaming Manager	**roam_pipeline**
Vertex Data Manager	**dm_vertex**
Webhook	**webhook**
Web Services Manager	**brm_wsm** **brm_wsm_wls** **brm_wsm_wl_init**

Downloading BRM Images from Oracle Software Delivery Website

To download BRM cloud native images, such as the Billing Care image, from the Oracle Software Delivery website:

1. Go to https://edelivery.oracle.com.
2. Sign in to the Oracle Software Delivery website using an Oracle account.
3. Search for and select **Oracle Communications Billing and Revenue Management Cloud Native Deployment Option 15.1.***x.x.x* and then click **Continue**.
4. Select the following and then click **Continue**:
    - Oracle Communications Billing and Revenue Management Cloud Native Deployment Option 15.1.*x.x.x*-CN
    - Oracle Communications Pricing Design Center Cloud Native Deployment Option 15.1.*x.x.x*-CN
    - Oracle Communications Elastic Charging Engine Cloud Native Deployment Option 15.1.*x.x.x*-CN
5. Accept the Oracle standard terms and restrictions and then click **Continue**.
6. Select the packages listed in Table 5-3 and then click **Download**.

    Each package is downloaded to a separate Zip file.
7. Extract the package files listed in Table 5-3 from each Zip file.

Table 5-3   BRM Cloud Native Packages and Package Files

BRM Package Name	Package File Name
Oracle Communications Cloud Native Batch Controller	**oc-cn-brm-batch-controller-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native Batch Pipeline	**oc-cn-brm-batch-pipeline-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native Billing Care	**oc-cn-billingcare-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native Billing Care REST API	**oc-cn-bcws-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native BRM Applications	**oc-cn-brm-apps-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native BRM Database Initializer	**oc-cn-brm-init-db-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native BRM REST Services Manager	**oc-cn-brm-rest-services-manager-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native BRM SDK	**oc-cn-brm-sdk-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native Business Operations Center	**oc-cn-boc-15.1.***x.x.x***.tar**
Oracle Communications Cloud Native Connection Manager	**oc-cn-brm-cm-15.1.***x.x.x***.tar**

Table 5-3 (Cont.) BRM Cloud Native Packages and Package Files

BRM Package Name	Package File Name
Oracle Communications Cloud Native Database Upgrade	**oc-cn-brm-upgrade-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Elastic Charging Engine Cloud Native Deployment Option	**oc-cn-ece-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Email Data Manager	**oc-cn-brm-dm-email-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Enterprise Application Integration Data Manager	**oc-cn-brm-dm-eai-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Enterprise Application Integration Java Server	**oc-cn-brm-eai-js-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Fusa Data Manager	**oc-cn-brm-dm-fusa-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Fusa Simulator	**oc-cn-brm-fusa-simulator-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Invoice Data Manager	**oc-cn-brm-dm-invoice-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Invoice Formatter	**oc-cn-brm-invoice-formatter-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Kafka Data Manager	**oc-cn-brm-dm-kafka-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native LDAP Data Manager	**oc-cn-brm-dm-ldap-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Oracle Database Manager	**oc-cn-brm-dm-oracle-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Performance Profiling Toolkit	**oc-cn-brm-perflib-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Pipeline Configuration Center	**oc-cn-pcc-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Pricing Design Center	**oc-cn-pdc-15.1.*x.x.x*.tar** **oc-cn-pdc-rsm-15.1.*x.x.x*.tar** **oc-cn-pdc-rsm-jars-15.0.*x.x.x*.tar.gz**
Oracle Communications Cloud Native Provisioning Data Manager	**oc-cn-brm-dm-prov-telco-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Rated Event Loader	**oc-cn-brm-rel-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Rated Event Manager	**oc-cn-brm-rem-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Real-Time Pipeline	**oc-cn-brm-realtime-pipeline-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Roaming Manager	**oc-cn-brm-roam-pipeline-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Vertex Data Manager	**oc-cn-brm-dm-vertex-15.1.*x.x.x*.tar**
Oracle Communications Cloud Native Webhook	**oc-cn-brm-webhook-15.1.*x.x.x*.tar**

Table 5-3 (Cont.) BRM Cloud Native Packages and Package Files

BRM Package Name	Package File Name
Oracle Communications Cloud Native Web Services Manager	**oc-cn-brm-wsm-15.1.***x.x.x***.tar** **oc-cn-brm-wsm-init-15.1.***x.x.x***.tar** **oc-cn-brm-wsm-wls-15.1.***x.x.x***.tar**

8. Load each package file as an image into the Podman system using the following command:

```
podman load --input fileName
```

   where *fileName* is the package file name listed in Table 5-3.

   For example, to load the Kafka DM image in the Podman system, enter this command:

```
podman load --input oc-cn-brm-dm-kafka-15.1.x.x.x.tar
```

If you use an internal registry to access images from different Kubernetes nodes, push the images from your local system to the registry server. For example, if the registry is identified by *RepoHost*:*RepoPort*, you'd push the Kafka DM image to the registry using the Podman CLI like this:

1. Tag the Kafka DM image with the registry server:

```
podman tag dm-kafka:15.1.x.x.x RepoHost:RepoPort/dm-kafka:15.1.x.x.x
```

2. Push the Kafka DM image to the registry server:

```
podman push RepoHost:RepoPort/dm-kafka:15.1.x.x.x
```

Pulling WebLogic Images for PDC, Billing Care, Billing Care REST API, and Business Operations Center

If you use PDC, Billing Care, Billing Care REST API, or Business Operations Center, pull the Oracle WebLogic image from the Oracle Container Registry into your private repository.

To load the Oracle WebLogic image into your private repository:

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

   Note:

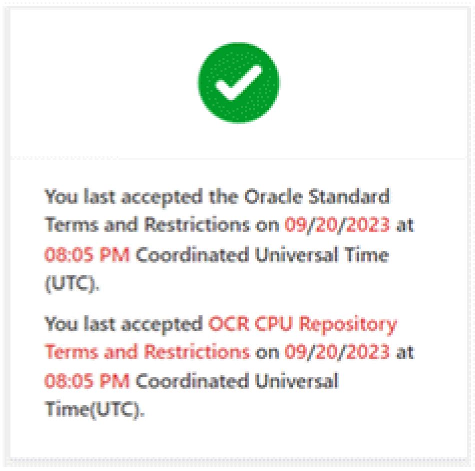
   To pull images for licensed software on the Oracle Container Registry, you must have an Oracle account. You can create an Oracle account at https://profile.oracle.com/myprofile/account/create-account.jspx.

3. Click the **Middleware** container, and then click the **fmw-infrastructure_cpu** CPU repository.

   The repository page appears.

4. Accept the Oracle terms and restrictions by clicking **Continue** and, on the next page, clicking **Accept**.

   If successful, you will see something similar to this:



You last accepted the Oracle Standard Terms and Restrictions on 09/20/2023 at 08:05 PM Coordinated Universal Time (UTC).

You last accepted OCR CPU Repository Terms and Restrictions on 09/20/2023 at 08:05 PM Coordinated Universal Time(UTC).

5. On the host system, log in to the Oracle Container Registry using the Podman CLI:

```
podman login container-registry.oracle.com
```

6. When prompted for a user name and password, enter your Oracle credentials.
7. Pull the WebLogic image into your local system using the following command:

```
podman pull container-registry.oracle.com/middleware/fmw-infrastructure_cpu:12.2.1.4-
jdk8-ol8
```

8. Tag the image with the registry server using the following command, where *RepoHost* and *RepoPort* are the host and port of your private repository:

```
podman tag container-registry.oracle.com/middleware/fmw-infrastructure_cpu:12.2.1.4-
jdk8-ol8 RepoHost:RepoPort/middleware/fmw-infrastructure_cpu:12.2.1.x-jdk8-ol8
```

9. Push the image to the registry server using the following command, where *RepoHost* and *RepoPort* are the host and port of your private repository:

```
podman push RepoHost:RepoPort/middleware/fmw-infrastructure_cpu:12.2.1.4-jdk8-ol8
```

Part III
Configuring and Deploying BRM Cloud Native

This part provides information about configuring and deploying Oracle Communications Billing and Revenue Management (BRM) in your cloud native environment. It contains the following chapters:

6

Deploying the BRM Database Schema

Learn how to deploy a new or existing database schema in the Oracle Communications Billing and Revenue Management (BRM) cloud native environment.

Topics in this document:

- Deploying BRM with a New Database Schema
- Deploying BRM with an Existing Schema

Deploying BRM with a New Database Schema

To deploy BRM with a new BRM and pipeline database schema:

1. If you have not already done so, create and configure your BRM database. See "Creating and Configuring Your BRM Database".
2. Create a new Kubernetes namespace for **oc-cn-init-db-helm-chart**:

   ```
   kubectl create namespace InitDbNameSpace
   ```

3. Create an **override-values.yaml** file for **oc-cn-init-db-helm-chart**, and add keys from Table 6-1.

   Note:

   This documentation uses the **override-values.yaml** file name for ease of use, but you can name the file whatever you want.

4. If you want the init-db Helm chart to create users and tablespaces for you, add the following keys to your file:

   Note:

   Skip this step for production systems or if you pre-created the database user and tablespaces.

   - **db.user**: The user name of the system administrator.
   - **db.password**: The password for the system administrator.
   - **db.port**: The port number of the database server.

   During deployment, the init-db Helm chart uses these values to validate the schema user names and tablespace names that you provide. If any schema users or tablespace names are missing, the init-db Helm chart creates them using the system administrator credentials.

5. If you want to deploy the BRM cloud native schema into a multischema database, do the following in your **override-values.yaml** file:
   a. Set the **ocbrm.db.skipPrimary** key to **false**.
   b. For each secondary schema in your system, add an **ocbrm.db.multiSchemas.secondary***N* block, where *N* is **1** for the first secondary schema, **2** for the next secondary schema, and so on.
   c. In each **ocbrm.db.multiSchemas.secondary***N* block, set the following keys:

      Note:

      If the **host**, **port**, and **service** keys are not defined, the secondary schema uses the same host name, service, and port number as the primary schema.

      - **deploy**: Set this to **true** to deploy this secondary schema.
      - **host**: Set this to the host name of the secondary schema. This key is optional.
      - **port**: Set this to the port number for the secondary schema. This key is optional.
      - **service**: Set this to the service name for the secondary schema. This key is optional.
      - **schemauser**: Set this to the schema user name.
      - **schemapass**: Set this to the schema password.
      - **schematablespace**: Set this to the name of the schema tablespace, such as pin01.
      - **indextablespace**: Set this to the name of the index tablespace, such as pinx01.
   d. Enable account migration between your schemas by setting the **ocbrm.isAmt** key to **true**.

      See "Understanding Account Migration" in *BRM Moving Accounts Between Database Schemas* for more information.

   This shows example entries for a BRM database with one primary schema, two secondary schemas, and account migration enabled:

```
ocbrm:
   isAmt: true
ocbrm:
   db:
      host: hostname
      port: 12345
      service: serviceName
      schemauser: pin01
      schemapass: password
      schematablespace: pin01
      indextablespace: pinx01
      skipPrimary: false
      multiSchemas:
         secondary1:
            deploy: true
            schemauser: pin02
            schemapass: password
            schematablespace: pin02
            indextablespace: pinx02
         secondary2:
            deploy: true
            schemauser: pin03
            schemapass: password
            schematablespace: pin03
            indextablespace: pinx03
```

6. To commit seed data to the BRM database, modify the init-db container (**configmap_create_obj_2.yaml**) to add the corresponding PCM_OP_CREATE_OBJ input flist. For example:

```
<PCM_OP $PIN_OPNAME=$PIN_CONF_INIT_OPNAME; $PIN_OPFLAGS=0>
0 PIN_FLD_POID           POID [0] $DB_NUMBER /config/recharge_card_type 0 0
0 PIN_FLD_NAME            STR [0] "-"
0 PIN_FLD_PROGRAM_NAME    STR [0] "load_pin_recharge_card_type"
0 PIN_FLD_HOSTNAME        STR [0] "-"
0 PIN_FLD_VERSION         STR [0] "1"
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
</PCM_OP>
```

7. Initialize the BRM database schema by running this command from the **helmcharts** directory:

```
helm install InitDbReleaseName oc-cn-init-db-helm-chart --namespace InitDbNameSpace
--values OverrideValuesFile
```

   where:

   - *InitDbReleaseName* is the release name for **oc-cn-init-db-helm-chart** and is used to track this installation instance.
   - *InitDbNameSpace* is the namespace for **oc-cn-init-db-helm-chart**.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-init-db-helm-chart**.

   The init-db Helm chart creates an init-db pod and job for each schema.

8. After the init-db Helm chart deploys successfully, delete the Helm chart from your namespace by running this command from the **helmcharts** directory:

```
helm delete InitDbReleaseName -n InitDbNameSpace
```

Table 6-1 lists the keys that directly impact the BRM database schema and pipeline database schema. Add these keys to your **override-values.yaml** file with the same path hierarchy.

Caution:

Some keys hold sensitive data. They must be handled carefully with controlled access to the file containing its values. Encode all of these values in Base64. See "Secrets" in *Kubernetes Concepts*.

Table 6-1 oc-cn-init-db-helm-chart Keys

Key	Purpose
imageRepository	The registry server where you have pushed images. Typically, in the format “RepoHost:RepoPort/”. The value is added as a prefix to all image names when you install or upgrade Helm charts. This key is empty by default.
imagePullSecrets	The name of the Secret containing credentials for accessing images from your private image server. This is added to each pod to give it permission to pull the image from your private registry server. See "Creating Secrets for Docker Registry Authorization" for more information.
uniPass	Use this key to apply a uniform password to all BRM cloud native services, including: • Database Schemas • BRM Root Login • Oracle Wallets • WebLogic User 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.
db.sslMode	The type of SSL connection required for connecting to the database: **TWO_WAY**, **ONE_WAY**, or **NO**. The default is **ONE_WAY**.
db.user **db.password** **db.port**	The system administrator credentials. Set these keys only if you want the init-db Helm chart to create database users and tablespaces for you. The init-db Helm chart creates any missing schema users and tablespaces using the system administrator credentials. • **db.user**: The user name for the system administrator. • **db.password**: The password for the system administrator. • **db.port**: The port number of the database server. **Note:** If the these keys are not included, the init-db Helm chart assumes that the database user and tablespaces have been pre-created and throws an error if it runs into a connection issue.
ocbrm.imagePullPolicy	Specify when to pull images: • **IfNotPresent**: Pulls an image only if one is not present locally. This is the default. • **Always**: Always pulls an image.
ocbrm.isAmt	Specify whether account migration is enabled in your BRM multischema database: • **True**: Account migration is enabled. • **False**: Account migration is disabled. This is the default.
ocbrm.isIPV6Enabled	Specify whether IPV6 is enabled in your Kubernetes environment: • **True**: IPV6 is enabled. • **False**: IPV6 is disabled. This is the default.
ocbrm.ece_deployed	Specify whether to deploy ECE in your BRM cloud native environment: • **True**: ECE configurations will be loaded into your environment. This is the default. • **False**: ECE configurations will not be loaded into your environment.

Table 6-1 (Cont.) oc-cn-init-db-helm-chart Keys

Key	Purpose
ocbrm.pdc_deployed	Specify whether to deploy PDC in your BRM cloud native environment: • **True**: PDC configurations will be loaded into your environment. • **False**: PDC configurations will not be loaded into your environment. This is the default.
ocbrm.existing_rootkey_wallet	Specify whether you are using an existing BRM database or an existing root key wallet: • **True**: Uses your existing BRM database or root key wallet. • **False**: Performs a fresh database initialization. This is the default. When set to **false**, the init-db Helm chart runs **lds-config-job** to load default strings into the BRM database during deployment. Manually delete **lds-config-job** after BRM deploys successfully.
ocbrm.is_upgrade	Specify whether to upgrade the existing database schema: • **True:** Upgrades your existing database schema. • **False:** Deploys a new BRM database schema. This is the default.
ocbrm.isSSLEnabled	Specify whether SSL is enabled in the BRM database. The default is **true**.
ocbrm.cmSSLTermination	Specify whether to disable SSL between the CM and DM/EM. • **true**: The CM is the SSL endpoint. Set it to **true** when using a custom TLS certificate for the CM. When set to **true**, also set the **isSSLEnabled** key to **true**. • **false**: SSL is enabled across all components, from client to CM to DMs and EMs. This is the default.
ocbrm.customSSLWallet	Whether to use a custom TLS certificate for the CM. • **true**: A custom TLS certificate is used for the CM. When set to **true**, also set the **cmSSLTermination** key to **true** and move the Oracle wallet (brm_custom_wallet) containing the custom TLS certificate to the top level of the Helm chart. • **false**: The default TLS certificate is used for the CM. This is the default.
ocbrm.root_key_rotate	Whether to rotate the BRM root key. The default is **false**. For more information, see "Rotating the BRM Root Key" in *BRM Cloud Native System Administrator's Guide*.
ocbrm.brm_root_pass	The BRM root password. The value can be per user. You must generate a Secret. Add this key to your **override-values.yaml** file with the same path hierarchy.
ocbrm.rotate_brm_role_passwords	Whether to rotate the BRM role passwords. The default is **false**.

Table 6-1 (Cont.) oc-cn-init-db-helm-chart Keys

Key	Purpose
ocbrm.brm_role_pass.*	The initial passwords for each BRM role. The roles grant users the permission to access different BRM components, such as Customer Center or Pipeline Configuration Center (PCC). **Note:** You must set all of these passwords when the **unipass** key is not set. You set passwords for the following roles: • **acct_recv.0.0.0.1**: Accounts receivable • **bc_client.0.0.0.1**: Billing Care • **bill_inv_pymt_sub.0.0.0.1**: Invoice payments • **billing.0.0.0.1**: Billing • **boc_client.0.0.0.1**: Business Operations Center • **collections.0.0.0.1**: Collections • **crypt_utils.0.0.0.1**: Encryption utilities • **cust_center.0.0.0.1**: Customer Center • **cust_mgnt.0.0.0.1**: Customer management • **invoicing.0.0.0.1**: Invoicing • **java_client.0.0.0.1**: Java clients • **load_utils.0.0.0.1**: Load utilities • **payments.0.0.0.1**: Payments • **pcc_client.0.0.0.1**: PCC • **rerating.0.0.0.1**: Rerating • **rsm.0.0.0.1**: BRM REST Services Manager • **super_user.0.0.0.1**: Super User • **ui_client.0.0.0.1**: All thick clients • **ece.0.0.0.1**: ECE **Note**: After you deploy or upgrade the database schema, you should rotate the role passwords regularly to improve security. For more information, see "Rotating BRM Role Passwords" in *BRM Cloud Native System Administrator's Guide*.
ocbrm.wallet.*	Specify the passwords for these wallets: • **client**: The password for the client wallet. • **server**: The password for the server wallet. • **root**: The password for the root wallet. You must generate Secrets for these keys.

Table 6-1 (Cont.) oc-cn-init-db-helm-chart Keys

Key	Purpose
ocbrm.db.*	The details for connecting to a shared database. The keys in this section take precedence over other database connection keys. Add these keys only if your system uses a shared database: • **host**: The host name of the machine on which the BRM database is configured. • **port**: The port on which the BRM database is configured. • **service**: The BRM database service name. • **sslMode**: The type of SSL connection required for connecting to the database: **TWO_WAY**, **ONE_WAY**, or **NO**. The default is **ONE_WAY**. • **walletPassword**: The password for accessing the database certificates from the TrustStore and KeyStore. • **walletType**: The type of file specified as the TrustStore or KeyStore for database SSL connections: **SSO** or **pkcs12**. • **enable_partition**: Whether partitioning is enabled in your BRM database: **Yes** or **No**. The default is **Yes**. • **storage_model**: The size of the BRM database tablespaces: **Test** (less than 700 MB), **Small** (less than 1.5 GB), **Medium** (less than 30 GB), or **Large** (greater than 30 GB). • **schemauser**: The user name for the primary BRM schema. • **schemapass**: The password for the database schema. • **schematablespace**: The name of the schema tablespace. • **indextablespace**: The name of the index tablespace. • **nls_lang**: The language, territory, and character set. Set this to **American_America.***characterset*, where *characterset* is either **UTF8** or **AL32UTF8**. **Note:** You must use **American_America** as the language and territory, regardless of your locale. • **pipelineschemauser**: The BRM pipeline schema user name. • **pipelineschemapass**: The BRM pipeline schema password. You must generate a Secret. Add this key to your **override-values.yaml** file with the same path hierarchy. • **pipelineschematablespace**: The name of the tablespace for the BRM pipeline schema. This field is case-sensitive. • **pipelineindextablespace**: The name of the index tablespace for the BRM pipeline schema. This field is case-sensitive. • **skipPrimary**: Whether to skip the deployment of the primary schema: **False** (deploy the primary schema) or **True** (do not deploy the primary schema). Set **skipPrimary** to **true** only if you are adding a schema to an existing BRM multischema system. See "Adding Schemas to a Multischema System" in *BRM Cloud Native System Administrator's Guide*.

Table 6-1 (Cont.) oc-cn-init-db-helm-chart Keys

Key	Purpose

ocbrm.db.multiSchemas.secondary*N*	The details for connecting to your secondary schemas, where *N* is **1** for the first secondary schema, **2** for the next secondary schema, and so on.
	Add this block only if your BRM database contains multiple schemas. This section will be commented out by default:
	`multiSchemas:`
	`   secondary1:`
	`      deploy: true`
	`      host: localhost`
	`      port: 1521`
	`      service: pindb`
	`      schemauser:`
	`      schemapass:`
	`      schematablespace:`
	`      indextablespace:`

Deploying BRM with an Existing Schema

To deploy BRM with an existing schema and with default Oracle images:

1. Copy the root-key wallet files from the **$BRM_WALLET/client** directory to the **oc-cn-helm-chart/existing_wallet** directory.
2. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **ocbrm.existing_rootkey_wallet** key to **true**.
3. Deploy **oc-cn-helm-chart**.

Alternatively, you could deploy BRM with an existing schema by doing this:

1. Create images for each BRM Server component in the installed BRM staging area (using the same staging area that initialized the database).
2. In your **override-values.yaml** file for **oc-cn-helm-chart**, update the keys with the existing schema credentials and also set the following keys:
   - **ocbrm.existing_rootkey_wallet:** Set this to **true**.
   - **ocbrm.use_oracle_brm_images:** Set this to **false**.
   - **ocbrm.db.queuename:** Set this to match the queue name in the existing database.
   - **obcrm.db.roamqueuename**: If you want a database queue for Roaming Pipeline, create another queue in the Oracle database by following the instructions in "Creating Additional Queues for Multischema BRM Systems" in *BRM Installation Guide*. Then, set the **roamqueuename** key to the name of the queue you created.
3. Deploy **oc-cn-helm-chart**.

7

Configuring BRM Server and PDC Services

Learn how to configure the Oracle Communications Billing and Revenue Management (BRM) server and Pricing Design Center (PDC) in your cloud native environment.

Topics in this document:

About Configuring BRM Cloud Native Services

You configure and deploy BRM cloud native services, such as BRM, PDC, and PCC, by using the BRM Helm chart (**oc-cn-helm-chart**). YAML descriptors in the **oc-cn-helm-chart/templates** directory use the **oc-cn-helm-chart/values.yaml** file for most of the values. The **values.yaml** file itself includes comments that describe each key. You can override the values by creating an **override-values.yaml** file.

Oracle recommends that you read the **values.yaml** file at least once to become familiar with all of the options available.

Creating Secrets for Docker Registry Authorization

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

Automatically pulling images from a private container 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 --namespace NameSpace
   ```

   where:

   - *SecretName* is the name of your Kubernetes Secret
   - *RegistryServer* is your private container registry's FQDN (*repoHost*:*repoPort*)

- *UserName* and *Password* are your private container registry's user name and password
- *NameSpace* is the namespace you will use for installing BRM Helm charts

For example:

```
kubectl create secret docker-registry cgbu-docker-registry --docker-
server=mydockerimages.com:2660/ --docker-username=xyz --docker-password=password --
namespace oms
```

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

```
imagePullSecrets:
    -name "SecretName1"
    -name "SecretName2"
```

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

```
imageRepository: "RegistryServer"
```

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

Managing Wallet and KeyStore Certificates

You can create wallet and KeyStore certificates for your BRM cloud native deployment in one of these ways:

- Pre-create all KeyStore certificates and wallets as Secrets in the Kubernetes cluster before you deploy BRM cloud native. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
- Have the BRM cloud native installer create the Kubernetes Secrets for you. In this case, you store the wallet files and KeyStore certificates in the cloud native Helm charts. During the Helm install or upgrade process, the KeyStores are created as Kubernetes Secrets, which eventually end up as Secrets in the Kubernetes cluster.

Configuring Global Values

Table 7-1 lists the keys that apply to all BRM components. To set or change the values, add them to your **override_values.yaml** file for **oc-cn-helm-chart**.

Table 7-1 Global Keys in Values.yaml File

Key	Description
imageRepository	The registry server where you have pushed images. Typically, in the format “RepoHost:RepoPort/”. The value is added as a prefix to all image names when you install or upgrade Helm charts. This key is empty by default.
imagePullSecrets	The name of the Secret that contains credentials for accessing images from your private image server. This is added to each pod to give it permission to pull the image from your private registry server. See "Creating Secrets for Docker Registry Authorization" for more information. This key is empty by default.

Table 7-1 (Cont.) Global Keys in Values.yaml File

Key	Description
uniPass	Use this key to apply a uniform password to all BRM cloud native services, including: • Database Schemas • BRM Root Login • BRM Role Passwords • Oracle Wallets • WebLogic User 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.
db.*	The details for connecting to a shared database. The keys in this section take precedence over other database connection keys. Add these keys only if your system uses a shared database: • **sslMode**: The type of SSL connection required for connecting to the database: – **TWO_WAY**: Two-way SSL authentication is required. In this case, both the client and server must authenticate each others identity. – **ONE_WAY**: One-way SSL authentication is required. In this case, the client must authenticate the server's identity. This is the default. – **NO**: SSL authentication is not required. • **extDBSSLWalletSecret**: The name of the Secret containing the SSL database wallet. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **host**: The host name or IP address of the database server. • **port**: The port number of the database server. • **user**: The user name of the database administrator. • **password**: The password of the database system administrator. • **serviceName**: The service name that identifies the database. • **role**: The role assigned to the DBA user. • **walletPassword**: The password for accessing the certificates from the TrustStore and KeyStore. This is required if **sslMode** is set to **ONE_WAY** or **TWO_WAY**. • **walletType**: The type of file specified as the TrustStore or KeyStore for SSL connections: **SSO** or **pkcs12**.
security.*	The details for setting security in BRM cloud native: • **tlsVersions**: The list of TLS versions supported. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**. • **java.overrideSecurityProperties**: Whether to override the default Java security property (**true**) or not (**false**). The default is **true**.
monitoring.prometheus.jmx_exporter.enable	Whether to enable the JMX exporter for Prometheus (**true**) or not (**false**). The default is **false**. See "Monitoring BRM Cloud Native Services" in *BRM Cloud Native System Administrator's Guide* for more information.

Table 7-1   (Cont.) Global Keys in Values.yaml File

Key	Description
monitoring.prometheus.operator.*	The details for monitoring BRM cloud native services using Prometheus: • **enable**: Whether to use Prometheus Operator (**true**) or standalone Prometheus (**false**). The default is **false**. • **namespace**: The namespace in which Prometheus Operator is deployed. The default is **prometheus**. • **release**: The release name for Prometheus Operator. The default is **prometheus**. See "Monitoring BRM Cloud Native Services" in *BRM Cloud Native System Administrator's Guide* for more information.

Specifying the BRM Services to Deploy

Some BRM cloud native services are enabled by default, while others are disabled. Ensure that your **override-values.yaml** file is set up to deploy the services that you want to include in your BRM cloud native environment.

BRM Cloud Native Services Enabled by Default

Table 7-2 lists the BRM cloud native services that are deployed by default. To exclude them from your deployment, set the keys to **false** in your **override-values.yaml** file for **oc-cn-helm-chart**.

Table 7-2   BRM Services Enabled By Default

BRM Service	override-values.yaml Key
Batch Pipeline	**ocbrm.batchpipe.isEnabled**
Billing Care	**ocbc.bc.isEnabled**
Billing Care REST API	**ocbc.bcws.isEnabled**
Business Operations Center	**ocboc.boc.isEnabled**
BRM REST Services Manager	**ocrsm.rsm.isEnabled**
Connection Manager	**ocbrm.cm.isEnabled**
Oracle Data Manager	**ocbrm.dm_oracle.isEnabled**
Pipeline Configuration Center	**ocpcc.pcc.isEnabled**
Pricing Design Center	**ocpdc.isEnabled**
Rated Event (RE) Loader Daemon	**ocbrm.rel_daemon.isEnabled**
Realtime Pipeline	**ocbrm.realtimepipe.isEnabled**

BRM Cloud Native Services Disabled By Default

Table 7-3 lists the BRM cloud native services that are *not* deployed by default. To include them in your BRM cloud native deployment, set the keys to **true** in your **override-values.yaml** file for **oc-cn-helm-chart**.

Table 7-3 BRM Services Disabled By Default

BRM Service	override-values.yaml Key
Batch Controller	**ocbrm.batch_controller.isEnabled**
Billing Care REST API SDK	**ocbc.bcws.sdk.isEnabled**
Billing Care SDK	**ocbc.bc.sdk.isEnabled**
BRM Apps Jobs	**ocbrm.brm_apps.job.isEnabled**
BRM SDK	**ocbrm.brm_sdk.isEnabled**
Email Data Manager	**ocbrm.dm_email.isEnabled**
Enterprise Application Integration (EAI) Data Manager	**ocbrm.dm_eai.isEnabled**
Invoicing Formatter	**ocbrm.formatter.isEnabled**
Invoicing Data Manager	**ocbrm.dm_invoice.isEnabled**
Kafka Data Manager	**ocbrm.dm_kafka.isEnabled**
LDAP Data Manager	**ocbrm.dm_ldap.isEnabled**
Paymentech Data Manager	**ocbrm.dm_fusa.isEnabled**
PDC REST Services Manager	**ocpdcrsm.isEnabled**
Provisioning Data Manager	**ocbrm.dm_prov_telco.isEnabled**
RE Loader Daemon Job	**ocbrm.rel_daemon.job.isEnabled**
RE Loader Manager Job	**ocbrm.rel_manager.job.isEnabled**
Rated Event Manager	**ocbrm.rem.isEnabled**
Roaming Pipeline	**ocbrm.roampipe.isEnabled**
Vertex Data Manager	**ocbrm.dm_vertex.isEnabled**
Webhook	**webhook.isEnabled**
Web Services Manager Standalone	**ocbrm.wsm.soap.isEnabled**
Web Services Manager with WebLogic	**ocbrm.wsm.deployment.weblogic.isEnabled**
Web Services Manager with TomCat	**ocbrm.wsm.deployment.tomcat.isEnabled**

Configuring the BRM Server

To configure the BRM server to run in your cloud native environment, you override the BRM server-specific keys in the **values.yaml** file for **oc-cn-helm-chart**. Table 7-4 lists the keys that directly impact BRM Server pods. Add these keys to your **override-values.yaml** file with the same path hierarchy.

Note:

You can optionally deploy a simple demonstration version of BRM cloud native by using the sample **override_values.yaml** file that is packaged with **oc-cn-helm-chart**. This sample override file contains the bare minimum keys that you need to update to create a simple BRM cloud native system with the following services enabled by default: Account Synchronization DM, Batch Pipeline, CM, Oracle DM, RE Loader, Realtime Pipeline, Billing Care, Billing Care REST API, Business Operations Center, and PDC.

Table 7-4 BRM Server Keys

Key	Path in values.yaml File	Description
isAmt	**ocbrm**	Whether account migration is enabled in your BRM database (**true**) or not (**false**). The default is **false**.
isIPV6Enabled	**ocbrm**	Whether IPV6 is enabled in your Kubernetes environment (**true**) or not (**false**). The default is **false**.
ece_deployed	**ocbrm**	Whether ECE is going to be deployed in your BRM cloud native environment (**true**) or not (**false**). The default is **true**.
pdc_deployed	**ocbrm**	Whether PDC is going to be deployed: • **true**: Configuration data is *not* loaded into the BRM database. Only mandatory configuration records are loaded into the BRM database for starting the realtime pipeline pod. For the batch pipeline and roaming pipeline pods, you must load the required configuration data using PDC before deploying the pods. • **false**: Configuration data is loaded into the BRM database during deployment. This is the default.
use_oracle_brm_images	**ocbrm**	Whether to use the default BRM images (**true**) or not (**false**). Set this to **false** if you are building custom images. The default is **true**.
existing_rootkey_wallet	**ocbrm**	Whether you are deploying with an existing database or using an existing root key wallet: • **true**: You are deploying with an existing database or are using an existing root key wallet. • **false**: You are deploying with a new database and are using a new root key wallet. This is the default. When set to **false**, the BRM Helm chart runs **lds-config-job** to load default strings into BRM during the deployment process. Manually delete **lds-config-job** after BRM is deployed successfully. See "Rotating the BRM Root Key" in *BRM Cloud Native System Administrator's Guide* for more information.
extExistingRootKeyWalletSecret	**ocbrm**	The name of the Kubernetes Secret containing the existing root key wallet for BRM Server. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
is_upgrade	**ocbrm**	Whether to upgrade the Helm chart (**true**) or not (**false**). See "Upgrading Your BRM Cloud Native Services".

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
isSSLEnabled	**ocbrm**	For SSL-enabled deployment required in **Infranet.properties**.
cmSSLTermination	**ocbrm**	Whether to make the CM the SSL endpoint for the BRM cloud native deployment: • **true**: The CM is the SSL endpoint. In this case, TLS can be enabled only between BRM client applications and the CM. TLS is disabled between CM and all downstream components such as DMs and EMs. Communication between external clients and the CM will still be encrypted by TLS 1.2. This setting can increase performance, because it eliminates the overhead needed to handle TLS before processing the PCP packets. When set to **true**, also set the **isSSLEnabled** key to **true**. • **false**: SSL is enabled across all components, from client to CM to DMs and EMs. This is the default.
customSSLWallet	**ocbrm**	Whether to use a custom TLS certificate for the CM: • **true**: A custom TLS certificate is used for the CM. When set to **true**, also set the **cmSSLTermination** key to **true** and move the Oracle wallet (**brm_custom_wallet**) containing the custom TLS certificate to the top level of the Helm chart. • **false**: The default TLS certificate is used for the CM. This is the default. See "Using a Custom TLS Certificate" in *BRM Cloud Native System Administrator's Guide*.
extCustomSSLWalletSecret	**ocbrm**	The name of the Kubernetes Secret containing the custom TLS certificate wallet for BRM Server. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
EnableSecurityContext	**ocbrm**	Whether to enable a security context in the cluster (**true**) or not (**false**). The default is **false**.
root_key_rotate	**ocbrm**	Whether to rotate the BRM root key (**true**) or not (**false**). The default is **false**. See "Rotating the BRM Root Key" in *BRM Cloud Native System Administrator's Guide*.
brm_root_pass	**ocbrm**	The root password. See "Rotating the BRM Root Password" in *BRM Cloud Native System Administrator's Guide*.
rotate_password	**ocbrm**	Whether to rotate the BRM root password: • **true**: The BRM root password is replaced with the one specified in the **new_brm_root_password** key. • **false**: The BRM root password is not changed. This is the default. See "Rotating the BRM Root Password" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
new_brm_root_password	**ocbrm**	The new BRM root password. Use this key only when **ocbrm.rotate_password** is set to **true**. See "Rotating the BRM Root Password" in *BRM Cloud Native System Administrator's Guide*.
rotate_brm_role_passwords	**ocbrm**	Whether to rotate the BRM role passwords. The default is **false**. See "Rotating BRM Role Passwords" in *BRM Cloud Native System Administrator's Guide*.
brm_role_pass.*	**ocbrm**	The passwords for each BRM role. The roles grant users the permission to access different BRM components, such as Customer Center or Pipeline Configuration Center (PCC). **Note:** You must set all of these passwords when the **unipass** key is not set. You set passwords for the following roles: • **acct_recv.0.0.0.1**: Accounts receivable • **bc_client.0.0.0.1**: Billing Care • **bill_inv_pymt_sub.0.0.0.1**: Invoice payments • **billing.0.0.0.1**: Billing • **boc_client.0.0.0.1**: Business Operations Center • **collections.0.0.0.1**: Collections • **crypt_utils.0.0.0.1**: Encryption utilities • **cust_center.0.0.0.1**: Customer Center • **cust_mgnt.0.0.0.1**: Customer management • **invoicing.0.0.0.1**: Invoicing • **java_client.0.0.0.1**: Java clients • **load_utils.0.0.0.1**: Load utilities • **payments.0.0.0.1**: Payments • **pcc_client.0.0.0.1**: PCC • **rerating.0.0.0.1**: Rerating • **rsm.0.0.0.1**: BRM REST Services Manager • **super_user.0.0.0.1**: Super User • **ui_client.0.0.0.1**: All GUI clients • **ece.0.0.0.1**: ECE The passwords in this key must match the passwords in **oc-cn-init-db-helm-chart**. See "Rotating BRM Role Passwords" in *BRM Cloud Native System Administrator's Guide*.
wallet.*	**ocbrm**	Specify the passwords for these wallets: • **client**: The password for the client wallet. • **server**: The password for the server wallet. • **root**: The password for the root wallet. You must generate Secrets for these keys.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
cm.*	**ocbrm**	The details for deploying the cm pod: • **isEnabled**: Whether to enable the CM. The default is **true**. • **deployment.replicaCount**: The number of replicas to create of the cm pod. The default is 1. • **deployment.imageName**: The name of the cm image. • **deployment.imageTag**: The tag name for the cm image. • **deployment.enable_publish**: Whether to publish events (**1**) or not (**0**). The default is **0**. • **deployment.enable_prefs_enrichment**: Whether to enrich notifications with subscriber preferences (**true**) or not (**false**). The default is **false**. • **deployment.prefs_enabled_publisher_list**: The list of publishers with enrichment enabled. The default is **0.0.9.6**. • **deployment.prefs_phone_no_location**: Where to retrieve the phone numbers for subscribers. The default is **0**. • **deployment.provisioning_enabled**: Whether to enable provisioning of service orders (**true**) or not (**false**). The default is **false**. • **deployment.simulate_agent**: Whether to publish service orders (**0**) or not (**1**). The default is **1**. • **deployment.perflib_enabled**: Whether to enable monitoring of the cm service using the performance library (Perlib). The default is **false**. See "Monitoring BRM Cloud Native Services" in *BRM Cloud Native System Administrator's Guide*. • **service.type**: The service type. The default is **ClusterIP**. • **service.serviceFqdn**: Set this to the CM's TLS certificate Subject Alternative Name, such as **dns:node1.brm.com**. • **custom_files.enable**: Whether to expose the **oc-cn-helm-chart/cm_custom_files** directory as a ConfigMap (**true**) or not (**false**). The default is **false**. The default is **false**. See "Exposing Directories as ConfigMaps" in *BRM Cloud Native System Administrator's Guide*. • **custom_files.extCustomFilesCM**: The name of the ConfigMap for the external CM custom files. • **resources.***: The minimum and maximum CPU and memory resources for the cm pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **hpaValues.***: The details for scaling up or down the number of pod replicas in your deployment based on the pod's CPU or memory utilization. By default, the Horizontal Pod Autoscaler is disabled. See "Setting Up Autoscaling of BRM Pods" in *BRM Cloud Native System Administrator's Guide*.
volume.storage	**ocbrm.custom_job_files**	The storage size of the volume. The default is **50Mi**.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
eai_js	**ocbrm**	The details for deploying the EAI Java Server. • **deployment.***: The details for deploying the EAI Java Server pod. • **extPayloadCM**: The name of the ConfigMap for the external payload files. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
dm_oracle.*	**ocbrm**	The details for deploying the dm-oracle pod: • **isEnabled**: Whether to enable the Oracle DM. The default is **true**. • **deployment**: The details for deploying the dm-oracle pod. • **config.totalFrontEnds**: The total number of front end processes. The default is **4**. • **config.totalBackEnds**: The total number of back end processes. The default is **4**. • **config.connectionsPerFrontEnd**: The number of connections for each front end process. The default is **16**. • **config.totalTransBackEnds**: The total number of back end transactions. The default is **4**. • **config.dmSequenceCacheSize**: The number of POIDs to cache when each instance of an Oracle DM is started. • **config.maxStatementCache**: The maximum size of the statement cache. • **config.sharedMemoryBigSize**: The size of the DM shared memory for "big" shared memory structures, such as those used for large searches. • **config.sharedMemorySegmentSize**: The size of the DM shared memory segment. • **secondaryConfig.***: The configuration for the secondary Oracle DM. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **hpaValues.***: The details for scaling up or down the number of pod replicas in your deployment based on the pod's CPU or memory utilization. See "Setting Up Autoscaling of BRM Pods" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
dm_kafka.*	**ocmbrm**	The details for configuring the Kafka DM: • **isEnabled**: Whether to enable the Kafka DM (**true**) or not (**false**). The default is **false**. • **kafkaAsyncMode**: Whether to use asynchronous mode, which logs business events that failed to publish (**true**) or synchronous mode, which returns errors for business events that failed to publish (**false**). The default is **false**. • **maxBlock**: The maximum block size. The default is **3000**. • **extKafkaKeystoreSecret**: The name of the Kubernetes Secret containing the KeyStore certificate files for Kafka DM. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **deployment.imageName**: The name of the dm_kafka image. The default is **dm_kafka**. • **deployment.imageTag**: The tag name for the dm_kafka image. • **deployment.replicaCount**: The number of replicas to create of the dm_kafka pod. The default is **1**. • **deployment.jvmOpts**: The Java options to configure, such as heap memory and JVM configurations. • **deployment.kafka_bootstrap_server_list**: Set this to a comma-separated list of addresses for the Kafka brokers in this format: *hostname1*:*port1*, *hostname*2:*port*2. • **deployment.poolSize**: Set this to the number of threads that can run in the JS server to accept requests from the CM. Enter a number from 1 through 2000. The default is **64**. • **deployment.topicName**: Set this to the name of the default Kafka topic. The default name is **BRMTopic**. • **deployment.topicFormat**: Set this to the format of the payload that is published to the default Kafka topic: **XML** or **JSON**. The default is **XML**. • **deployment.topicStyle**: The style of XML payloads: **ShortName**, **CamelCase**, **NewShortName**, or **OC3CNotification**. The default is **CamelCase**. • **deployment.isSecurityEnabled**: Whether to enable SSL between Kafka DM and Kafka Server (**true**) or not (**false**). The default is **false**. • **deployment.trustStorePassword**: The TrustStore password in Base64 format. Required only if SSL is enabled. • **deployment.keyStorePassword**: The KeyStore password in Base64 format. Required only if SSL is enabled. • **deployment.keyPassword**: The password of a key in the KeyStore in Base64 format. Required only if SSL is enabled. • **deployment.password**: The password in Base64 format. Required only if SSL is enabled. • **volume.storage**: The storage size of the volume.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
		• **volume.createOption**: By default, the Kafka DM uses dynamic volumes. To use a static volume instead, add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **hpaValues.***: The details for scaling up or down the number of pod replicas in your deployment based on the pod's CPU or memory utilization. See "Setting Up Autoscaling of BRM Pods" in *BRM Cloud Native System Administrator's Guide*. For more information about integrating BRM cloud native with a Kafka Server, see "Integrating with Kafka Servers" in *BRM Cloud Native System Administrator's Guide*.
dm_email.*	**ocbrm**	The details for configuring the Email DM: • **isEnabled**: Whether to enable the email DM. The default is **false**. • **deployment.***: The details for deploying the dm-email pod. • **config.totalFrontEnds**: The total number of front end processes. The default is **4**. • **config.totalBackEnds**: The total number of back end processes. The default is **4**. • **config.connectionsPerFrontEnd**: The number of connections for each front end process. The default is **16**. • **config.totalTransBackEnds**: The total number of back end transactions. The default is **4**. • **config.dmSequenceCacheSize**: The number of POIDs to cache when each instance of an Email DM is started. • **config.maxStatementCache**: The maximum size of the statement cache. • **config.sharedMemoryBigSize**: The size of the DM shared memory for "big" shared memory structures, such as those used for large searches. • **config.sharedMemorySegmentSize**: The size of the DM shared memory segment. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
dm_invoice.*	**ocbrm**	The details for configuring the Email DM: • **isEnabled**: Whether to enable the Invoice DM. The default is **false**. • **deployment.***: The details for deploying the dm-invoice pod. • **config.totalFrontEnds**: The total number of front end processes. The default is **2**. • **config.totalBackEnds**: The total number of back end processes. The default is **6**. • **config.connectionsPerFrontEnd**: The number of connections for each front end process. The default is **16**. • **config.totalTransBackEnds**: The total number of back end transactions. The default is **4**. • **config.dmSequenceCacheSize**: The number of POIDs to cache when each instance of an Email DM is started. • **config.maxStatementCache**: The maximum size of the statement cache. • **config.sharedMemoryBigSize**: The size of the DM shared memory for "big" shared memory structures, such as those used for large searches. • **config.sharedMemorySegmentSize**: The size of the DM shared memory segment. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
dm_prov_telco.*	**ocbrm**	The details for configuring Provisioning DM: • **isEnabled**: Whether to enable the Provisioning DM. The default is **false**. • **deployment.***: The details for deploying the Provisioning DM. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the dm-prov-telco pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
smtpServer	**ocbrm.dm_email.deployment**	Set this to your SMTP server name, such as ocbrm.us.example.com.
create	**ocbrm.storage_class**	Whether to create a Kubernetes StorageClass (**true**) or not (**false**).

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
virtual_time.*	**ocbrm**	The details for configuring the **pin_virtual_time** utility: • **enabled**: Set this to **true** to enable **pin_virtual_time**. • **sync_pvt_time**: Set this to the number of seconds between each synchronization of **pin_virtual_time** with all pods. The default is **0** seconds. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the virtual-time pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
db.*	**ocbrm**	The details for connecting to the BRM database: • **host**: The host name or IP address of the database server. • **port**: The port number of the database server. • **service**: The service name that identifies the database. • **sslMode**: The type of SSL connection required for connecting to the database: **TWO_WAY, ONE_WAY**, or **NO**. • **extDBSSLWalletSecret**: The name of the Kubernetes Secret containing the SSL database wallet. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **walletPassword**: The password for accessing the certificates from the TrustStore and KeyStore. This is required if **sslMode** is set to **ONE_WAY** or **TWO_WAY**. • **walletType**: The type of file specified as the TrustStore or KeyStore for SSL connections: **SSO** or **pkcs12**. The default is **SSO**. • **enable_partition**: Whether partitioning is enabled at the database level (**Yes**) or disabled (**No**). The default is **Yes**. • **storage_model**: The size of the BRM database tablespaces: – **Test**: Less than 700 MB. – **Small**: Less than 1.5 GB. – **Medium**: Less than 30 GB. – **Large**: Greater than 30 GB. • **schemauser**: The user name of the primary BRM schema. The default is **pin**. • **schemapass**: The password for the BRM schema. • **schematablespace**: The name of the tablespace for the primary BRM schema. This field is case-sensitive. The default is **pin**. • **indextablespace**: The name of the index tablespace for the primary BRM schema. This field is case-sensitive. The default is **pinx**. • **nls_lang**: The language, territory, and character set. Set this to **American_America.AL32UTF8**. You must use **American_America** as the language and territory, regardless of your locale. • **pipelineschemauser**: The BRM pipeline schema user name, which should be pre-created with all of the required grants. • **pipelineschemapass**: The BRM pipeline schema password. You must generate a Secret. Add this key to your **override-values.yaml** file with the same path hierarchy. • **pipelineschematablespace**: The name of the tablespace for the BRM pipeline schema. This field is case-sensitive. • **pipelinesindextablespace**: The name of the index tablespace for the BRM pipeline schema. This field is case-sensitive.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
		• **skipPrimary**: Whether to deploy the primary schema (**false**) or skip the deployment of the primary schema (**true**). Set it to **true** only if you are adding a schema to an existing BRM multischema system. The default is **false**. Ensure these values match the **ocbrm.db.*** keys from **oc-cn-init-db-helm-chart**. See Table 6-1 for more information.

secondary*N*.*	**ocbrm.db.multiSchemas**	The details for connecting to your secondary database schemas, where *N* is **1** for the first secondary schema, **2** for the next secondary schema, and so on.
		Add this block only if your BRM database contains multiple schemas. This section will be commented out by default:
		`multiSchemas:`
		`   secondary:`
		`      deploy:`
		`      host: localhost`
		`      port: 1521`
		`      service: pindb`
		`      schemauser:`
		`      schemapass:`
		`      schematablespace:`
		`      indextablespace:`
		See "Configuring BRM for a Multischema Database".

mountOptions	**ocbrm.storage_class**	Set this to the version of the external provisioner.
provisioner	**ocbrm.dynamic_provisioner**	Set this to the name of the external provisioner.
dm_fusa.*	**ocbrm**	The details for configuring the Paymentech DM: • **isEnabled**: Whether to enable the Paymentech DM. The default is **false**. • **deployment.***: The details for deploying the Paymentech DM. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the dm-fusa pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
realtimepipe.*	**ocbrm**	The details for configuring the Realtime Pipeline: • **isEnabled**: Whether to enable the Realtime Pipeline (**true**) or not (**false**). The default is **true**. • **deployment.replicaCount**: The number of replicas to create of the realtimepipe pod. The default is **1**. • **deployment.rtp_num_thread**: The number of threads in the realtime pipeline. The default is **8**. • **deployment.rtp_num_pipe**: The number of realtime pipelines. The default is **2**. • **deployment.discount_trace**: Whether to generate a discount trace file. The default is **true**. • **deployment.SemaphoreEnable**: Whether to check for and process semaphore files, which allow you to configure and control Pipeline Manager during runtime. The default is **true**. For more information, see "Using Semaphore Files to Control Pipeline Manager" in *BRM Pipeline Manager Administration Guide*. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the realtimepipe pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
batch_controller.*	**ocbrm**	The details for configuring the Batch Controller: • **isEnabled**: Whether to enable the BRM Controller. The default is **false**. • **deployment.***: The details for deploying the Batch Controller. • **volume.***directory***.createOption**: By default, all of the pipelines in the batch-controller pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key under the *directory* key, where *directory* is **input**, **archive**, and **reject**. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
rel_daemon.*	**ocbrm**	The details for configuring the Rated Event (RE) Loader daemon: • **isEnabled**: Whether to enable the RE Loader daemon. The default is **true**. • **job.isEnabled**: Whether to run the RE Loader daemon job. The default is **false**. • **deployment.***: The details for deploying the RE Loader daemon. • **volume.***directory***.createOption**: By default, all of the directories in the rel-daemon pod use dynamic volumes. To use static volumes instead, you must add the **createOption** key under the *directory* key, where *directory* is **reject**, **archive**, and **input**. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
rem.*	**ocbrm**	The details for configuring the rated-event-manager pod. **Note**: If **ocbrm.db.sslMode** is enabled (**ONE_WAY** or **TWO_WAY**), you must specify the server certificate's distinguished name (DN) in the **oc-cn-helm-chart/templates/configmap_rem_properties.yaml** file's **rated_event_manager.jdbc_pool.sslServerCertDN** key. • **logging.***: The details for logging information about the rated-event-manager pod. • **isEnabled**: Whether to deploy the rated-event-manager pod. The default is **false**. • **deployment.***: The details for deploying the rated-event-manager pod, such as its image name, its image tag, and the Kafka topic to use for rated events. • **volume.***directory***.createOption**: By default, all of the directories in this pod use dynamic volumes. To use a static volume instead, you must add the **createOption** key under the *directory* key, where *directory* is **input**, **reject**, **archive**, and **data**. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. • **volume.***directory***.storage**: The storage size of the volume. • **nodeSelector**: The rules for deploying the rated-event-manager pod on specific nodes. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **hpaValues.***: The details for scaling up or down the number of pod replicas in your deployment based on the pod's CPU or memory utilization. By default, the Horizontal Pod Autoscaler is disabled. See "Setting Up Autoscaling of BRM Pods" in *BRM Cloud Native System Administrator's Guide*.
batchpipe.*	**ocbrm**	The details for configuring the batch pipeline. • **isEnabled**: Whether to enable the batch pipeline. The default is **true**. • **deployment.***: The details for deploying the batch pipeline. • **volume.output.storage**: The storage size of the volume. • **volume.***directory***.createOption**: By default, all of the directories in the batchpipe pod use dynamic volumes. To use a static volume instead, you must add the **createOption** key under the *directory* key, where *directory* is **data**, **output**, **reject**, and **log**. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
roampipe.*	**ocbrm**	The details for configuring the roaming pipeline: • **isEnabled**: Whether to enable the roaming pipeline. The default is **false**. • **deployment.***: The details for deploying the roaming pipeline. • **volume.***directory***.createOption**: By default, all of the pipelines in the roampipe pod use dynamic volumes. To use a static volume instead, you must add the **createOption** key under the *directory* key, where *directory* is **output** and **reject**. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
cmt.*	**ocbrm**	The details for configuring and running **pin_cmt**. • **enabled**: Set this to **true** to run the Conversion Manager **pin_cmt** utility. The default is **false**. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the cmt pod uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
config_jobs.*	**ocbrm**	The details for running a configurator job, which allows you to run BRM load utilities on demand without entering into a pod: • **deployment.***: The details for deploying the configurator job. • **run_apps**: Set to **true** to enable a configurator job. The default is **false**. • **isMultiSchema**: Specifies whether to run the commands in the **loadme.sh** script on the secondary schemas. The default is **false**. • **restart_count**: Increment this count by 1 to restart the CM. • **script_name**: The name of the script that contains the load utilities you want to run. The default is **loadme.sh**. • **configmap_path**: The directory in which ConfigMaps are stored. The default is **/oms/load**. • **extCustomScriptsCM**: The name of the ConfigMap containing custom scripts to run. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. See "Running Load Utilities through Configurator Jobs" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
brm_apps.*	**ocbrm**	The details for running a brm-apps job, which allows you to run BRM applications and utilities on demand without entering into a pod: • **job.isEnabled**: Set to **true** to enable a brm-apps job. • **isMultiSchema**: Specifies whether to run the commands in the **loadme.sh** script on the secondary schemas. The default is **false**. • **job.configmap_path**: The path to the ConfigMap file. • **job.script_name**: The name of the script that contains the utilities and applications you want to run. The default is **loadme.sh**. • **extCustomScriptsCM**: The name of the ConfigMap containing custom scripts to run. • **deployment.***utilityName***.***: The configuration details for running BRM utilities and applications, such as **pin_billd**, **pin_export_price**, and **pin_rerate**. See "Running Applications and Utilities through BRM-Apps Jobs" in *BRM Cloud Native System Administrator's Guide*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
wsm.soap.*	**ocbrm**	Details about the standalone Web Services Manager service. • **isEnabled**: Whether to enable SOAP-based Web Services Manager services. The default is **false**. • **deployment.***: The details for deploying the brm-wsm pod. • **service.type.***: The service type. The default is **ClusterIP**. • **service.resources.***: The minimum and maximum CPU and memory resources that brm-wsm can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **configEnv.port**: The HTTP port where Web Services Manager is exposed. The default is **8080**. • **configEnv.httpsPort**: The HTTPS port where Web Services Manager is exposed. The default is **8443**. • **configEnv.inputValidationEnabled**: Whether to validate the input XML payload. The default is **true**. • **configEnv.soapInputValidationReportOnly**: Whether the validation errors in the input XML payload are logged to the console (**true**) or sent back as a fault only (**false**). The default is **false**. • **configEnv.outputValidationEnabled**: Whether to validate the output XML payload. The default is **true**. • **configEnv.soapOutputValidationReportOnly**: Whether the validation errors in the output XML payload are logged to the console (**true**) or sent back as a fault only (**false**). The default is **false**. • **configEnv.logLevel**: The logging level. The default is **INFO**. • **configEnv.tlsEnabled**: Whether to enable TLS encryption for Web Services Manager. The default is **false**. • **configEnv.externalSecretName**: The name of the Kubernetes Secret containing the passwords for standalone Web Services Manager. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **configEnv.clientAuth**: Whether OAuth 2.0 tokens in client requests are required (**REQUIRED**), optional (**OPTIONAL**), or not required (**NONE**). • **configEnv.jvmOpts**: The Java options to configure, such as heap memory and JVM configurations. • **configEnv.keyStoreFileName**: The file name of the KeyStore. • **configEnv.keyStoreAlias**: The private key alias of the KeyStore. • **configEnv.trustStoreFileName**: The file name of the TrustStore. • **configEnv.isOauthEnabled**: Whether OAuth 2.0 is enabled in the standalone version of Web Services Manager. The default is **false**.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
		• **configEnv.oauthCertificateName**: The name of the OAuth certificate file. • **configEnv.outputDateFormat**: The format for the prefix UNIX time stamp of the output XML payload file. • **configEnv.outputPrefixUnixTimestamp**: Whether to add a UNIX time stamp prefix to the output XML payload file name. The default is **false**. • **configEnv.inputDateFormat**: The format for the prefix UNIX time stamp of the input XML payload file. • **configEnv.inputPrefixUnixTimestamp**: Whether to add a UNIX time stamp prefix to the input XML payload file name. The default is **false**. • **configEnv.outputNamespacePrefixSoap**: The prefix to add to SOAP XML elements in the response. The default is **S**. • **configEnv.outputNamespacePrefixPayload**: The prefix to add to the payload XML elements in the response. The default is **brm**. • **configEnv.tracing.***: The details for tracing the BRM-WSM service. See "Using Metrics and Tracing (Standalone only)" in *BRM Web Services Manager*. • **secrets.***: The name of the Secrets containing the KeyStore and TrustStore passwords. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. For more information about Web Services Manager, see "Using Web Services" in *BRM Web Services Manager*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
wsm.deployment. weblogic.*	**ocbrm**	Details about Web Services Manager on WebLogic Server. • **isEnabled**: Whether to deploy Web Services Manager on WebLogic Server. The default is **false**. • **imageName**: The image name for Web Services Manager deployed on WebLogic Server. • **initImageName**: The name of the init image. • **imageTag**: The tag name for the Web Services Manager image. • **username**: The user name for accessing Web Services Manager. • **password**: The password for the user. • **replicaCount**: The number of replicas to create of the pod. The default is **1**. • **adminServerNodePort**: The NodePort where the admin-server's HTTP service will be accessible. By default, this key is blank. **Note**: Set this key only if you want the **brmdomain-admin-server-ext** service to deploy as NodePort. • **log_enabled**: Whether to create log files. The default is **false**. • **minPoolSize**: The minimum number of connections the connection pool can create. The default is **1**. • **maxPoolSize**: The maximum number of connections the connection pool can create. The default is **8**. • **poolTimeout**: The maximum amount of time in milliseconds that a connection request is queued. The default is **30000**. • **jvmOpts**: The Java options to configure, such as heap memory and JVM configurations. • **userMemArgs**: The memory arguments. • **serverStartPolicy**: The WebLogic servers that the Operator starts when it discovers the domain: **NEVER**, **ADMIN_ONLY**, and **IF_NEEDED**. • **monitoring.***: The details for collecting metrics for Web Services Manager deployed on WebLogic Server. See "Monitoring BRM Cloud Native Services" in *BRM Cloud Native System Administrator's Guide*. • **idp.***: The details about the Identity Provider (IdP) managing OAuth 2.0 tokens for authenticating clients to access BRM Web Services. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. For more information about Web Services Manager, see "Using Web Services" in *BRM Web Services Manager*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
wsm.deployment. tomcat.*	**ocbrm**	Details about Web Services Manager on Tomcat. • **isEnabled**: Whether to deploy Web Services Manager on Tomcat. The default is **false**. • **replicaCount**: The number of replicas to create of the pod. The default is **1**. • **imageName**: The image name for Web Services Manager deployed on Tomcat. • **imageTag**: The tag name for the Web Services Manager image. • **port**: The port number for Web Services Manager. The default is **8080**. • **nodePort**: The NodePort where the admin-server's HTTP service is accessible. The default is **30080**. This key applies only if **service.type** set to **NodePort**. • **log_enabled**: Whether to create log files. The default is **true**. • **minPoolSize**: The minimum number of connections the connection pool can create. The default is **1**. • **maxPoolSize**: The maximum number of connections the connection pool can create. The default is **8**. • **poolTimeout**: The maximum amount of time in milliseconds that a connection request is queued. The default is **30000**. • **basicAuth**: Whether an OAuth 2.0 authentication token is required. The default is **false**. • **walletPassword**: The passwords for the Web Services Manager wallet. • **extBasicConfigCM**: The name of the external ConfigMap containing the basic configuration files. • **service.type**: The service type. The default is **ClusterIP**. **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. For more information about Web Services Manager, see "Using Web Services" in *BRM Web Services Manager*.

Table 7-4 (Cont.) BRM Server Keys

Key	Path in values.yaml File	Description
brm_sdk.*	**ocbrm**	The details for the brm-sdk pod: • **isEnabled**: Whether to enable the BRM SDK. The default is **false**. • **extCustomScriptsCM**: The name of the external ConfigMap containing your custom scripts. • **deployment.***: The details for deploying the BRM SDK. • **volume.storage**: The storage size of the volume. • **volume.createOption**: By default, the brm-sdk pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. • **resources.***: The minimum and maximum CPU and memory resources for the pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Configuring BRM for a Multischema Database

Using a BRM multischema database lets you distribute customer accounts among several database schemas, providing increased storage capacity, higher performance, and easier maintenance. For more information, see "A BRM Multischema Production System" in *BRM Installation Guide*.

To configure your BRM cloud native environment to connect to a multischema database, do this:

1. Ensure that you deployed a multischema database in your BRM cloud native environment. See "Deploying BRM with a New Database Schema".
2. Using SQL*Plus, grant each database schema the privilege to insert and update tables on the other schemas:
   a. Connect to the BRM database with SQL*Plus as **sysdba**:

      ```
      sqlplus system@databaseAlias as sysdba
      Enter password: password
      ```

      where:

      - *databaseAlias* is the Oracle system database alias.
      - *password* is the Oracle system database user password.

   b. From the primary database schema, enter the following:

      ```
      SQL> GRANT INSERT ANY TABLE TO SecondarySchema;
      SQL> GRANT UPDATE ANY TABLE TO SecondarySchema;
      ```

      where *SecondarySchema* is the name of the secondary schema.

c. From each secondary database schema, enter the following:

```
SQL> GRANT INSERT ANY TABLE TO PrimarySchema;
SQL> GRANT UPDATE ANY TABLE TO PrimarySchema;
```

where *PrimarySchema* is the name of the primary schema.

3. Connect the BRM server to each secondary schema:
   a. Open your **override-values.yaml** file for **oc-cn-helm-chart**.
   b. Enable account migration by setting the **ocbrm.isAmt** key to **true**.
   c. Set the **ocbrm.db.skipPrimary** key to **false**.
   d. For each secondary schema in your system, add a **ocbrm.db.multiSchemas.secondary***N* block, where *N* is **1** for the first secondary schema, **2** for the next secondary schema, and so on.
   e. In each **ocbrm.db.multiSchemas.secondary***N* block, set the following keys:
      - **deploy**: Set this to **true**.
      - **host**: Set this to the hostname of the secondary schema. This key is optional.
      - **port**: Set this to the port number for the secondary schema. This key is optional.
      - **service**: Set this to the service name for the secondary schema. This key is optional.
      - **schemauser**: Set this to the schema user name.
      - **schemapass**: Set this to the schema password.
      - **schematablespace**: Set this to the name of the schema tablespace, such as pin01.
      - **indextablespace**: Set this to the name of the index tablespace, such as pinx01.
   f. Deploy **oc-cn-helm-chart** by running this command from the **helmcharts** directory:

      ```
      helm install BrmReleaseName oc-cn-helm-chart --namespace BrmNameSpace --values
      OverrideValuesFile
      ```

      where:
      - *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance. It must be different from the one used for **oc-cn-init-db-helm-chart**.
      - *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
      - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

      The BRM Helm chart deploys new dm-oracle, amt, and rel-dameon pods, Rated Event (RE) Loader PVCs, services, ConfigMaps, and secrets. It also updates their corresponding schema entries in the primary CM and Oracle DM and deploys multiple containers for the batch-wireless-pipe pod.
4. Set each database schema's status and priority. BRM cloud native assigns accounts to an open schema with the highest priority.
   a. Open the **configmap_pin_conf_testnap.yaml** file.

b. Under the **config_dist.conf** section, add the following entries for each secondary schema in your database:

```
DB_NO = "schema_number" ;              # database config. block
PRIORITY = priority ;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "status" ;
SCHEMA_NAME = "schema_name" ;
```

c. Set the **STATUS** and **PRIORITY** entries for each primary and secondary schema:

```
DB_NO = "0.0.0.1" ;              # Primary schema configuration block
PRIORITY = priority;
MAX_ACCOUNT_SIZE = 100000 ;
STATUS = "status" ;
SCHEMA_NAME = "schema_name" ;

DB_NO = "0.0.0.2" ;              # Secondary schema configuration block
PRIORITY = priority;
MAX_ACCOUNT_SIZE = 50000 ;
STATUS = "status" ;
SCHEMA_NAME = "schema_name" ;
```

where:

- *priority* is a number representing the schema's priority, with the highest number having the most priority. For example, 5 indicates a greater priority than a value of 1. For more information, see "Modifying Database Schema Priorities" in *BRM Cloud Native System Administrator's Guide*.
- *status* specifies whether the schema is **open**, **closed**, or **unavailable**. For more information, see "Modifying Database Schema Status" in *BRM Cloud Native System Administrator's Guide*.

d. Set up the configurator job to run the **load_config_dist** utility by adding the following lines to the **oc-cn-helm-chart/config_scripts/loadme.sh** script:

```
#!/bin/sh

#cp /oms/config_dist.conf /oms/sys/test/config_dist.conf
cd /oms/sys/test ; load_config_dist
exit 0;
```

e. In the **override-values.yaml** file for **oc-cn-helm-chart**, set this key:

ocbrm.config_jobs.run_apps: Set this to **true**.

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

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace BrmNameSpace
```

The distribution information is loaded into the primary schema.

g. Update these keys in the **override-values.yaml** file for **oc-cn-helm-chart**:

- **ocbrm.config_jobs.restart_count**: Increment the existing value by 1.
- **ocbrm.config_jobs.run_apps**: Set this to **false**.

h. Update the Helm release again:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace BrmNameSpace
```

The CM pod is bounced back.

5. Configure the account-router Pipeline Manager to route CDRs to pipelines based on the database schema POID. To do so, edit the ConfigMap file **configmap_acc_router_reg.yaml**.

   Based on the configuration, the account router Pipeline Manager does the following:

   - Moves input files to the **data** PVC directory. The input file names have a prefix of **router** and a suffix of **.edr**.
   - Moves the rated output files to the input of the Rating pipeline.
   - Replicates the Rating pipeline based on the multischema entry. The Range function is used to replicate the rating pipeline.
   - Moves the output files from the Rating pipeline to the **outputcdr** PVC directory.

Your BRM cloud native environment is connected to your BRM multischema database. To manage your multischema environment, see "Managing a BRM Cloud Native Multischema System" in *BRM Cloud Native System Administrator's Guide*.

Configuring Pricing Design Center

Pricing Design Center (PDC) is a Web-based client application that you use to create and manage the product offerings that you sell to your customers. A product offering represents the services available to your customers and the price of those services. For more information about PDC, see *Pricing Design Center Online Help*.

You can optionally deploy a simple demonstration version of Pricing Design Center cloud native by using the sample **PDC_OverrideValues.yaml** file provided with **oc-cn-helm-chart**. This simple demonstration version has both SSL and ECE enabled, uploads a sample JKS certificate file, loads sample RUMs and balance elements, and starts the BRM-to-PDC synchronization process with **SyncPDC**.

To configure PDC to run in your BRM cloud native environment:

1. Override the PDC-specific keys in the **values.yaml** file for **oc-cn-helm-chart**. See "Adding PDC Keys for oc-cn-helm-chart".
2. Override the PDC-specific keys in the **values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding PDC Keys for oc-cn-op-job-helm-chart".
3. Set up SAML for SSO in PDC. See "Setting Up SSO for PDC Cloud Native".

After you deploy PDC in your cloud native environment, you can access the PDC GUI at one of the following URLs:

- **`http://`***`kubernetesHost`*`:`*`pdcPort`***`/pdc`**

  where:

  – *kubernetesHost* is the host name of the machine on which Kubernetes is deployed.

  – *pdcPort* is the PDC service node port.

- **`http://`***`LoadBalancerHost`*`:`*`pdcNodePort`***`/pdc`**

  where:

  – *LoadBalancerHost* is the host name of the machine on which the load balancer is deployed.

  – *pdcNodePort* is the number assigned to the PDC node port.

Adding PDC Keys for oc-cn-helm-chart

Table 7-5 describes the most common PDC keys that you need to override. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart** with the same path hierarchy.

For information about all PDC-specific keys, see the descriptions in the **oc-cn-helm-chart/ values.yaml** file.

Caution:

Keys with the path **ocpdc.secretValue** hold sensitive data. Handle them carefully with controlled access to the file containing their values. Encode all of these values in Base64. See "Secrets" in *Kubernetes Concepts*.

Table 7-5 Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
isEnabled	**ocpdc**	Whether to enable and deploy PDC: • **true**: Enables PDC and deploys the PDC application. This is the default. • **false**: Disables the PDC application.
lang	**ocpdc**	The Linux system locale. The default is **en_US.UTF-8**.
tz	**ocpdc**	The Linux time zone. The default is **UTC**.
volume.createOption.*	**ocpdc**	By default, PDC uses dynamic volume provisioning. To use a static volume instead, add the **createOption** keys under the **volume** section. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. **Note**: Ensure you provide the required permissions to the volume path by following the guidelines in "Persistent Volume Storage Locations" in the WebLogic Kubernetes Operator documentation.
storageSize	**ocpdc**	The storage size for the pdc-brm-pv or pdc-brm-pvc, such as 10Gi.
enableSecurityContext	**ocpdc**	Whether to enable PDC pod-level security attributes and common container settings. The default is **true**.
deployment.*	**ocpdc**	The details for deploying the PDC image: • **imageName**: The name of the PDC image. The default is **oracle/pdc**. • **imageTag**: The tag name for the PDC image. The default is **:15.1.0.0.0**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.

Table 7-5 (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
deployment.fmw.*	**ocpdc.deployment**	The details for pulling WebLogic images for PDC from a container registry: • **imageRepository**: The name of the container registry from which to pull the WebLogic image. The default is **container-registry.oracle.com/**. • **imageName**: The name of the container repository from which to pull the WebLogic image. The default is **middleware/fmw-infrastructure**. • **imageTag**: The tag name for the WebLogic image. See "Pulling WebLogic Images for PDC, Billing Care, Billing Care REST API, and Business Operations Center" for more information.
nodeSelector	**ocpdc**	The name of the node on which to run the following PDC pods: • pdc • RRE • BRE • SyncPDC • Import-Export Set this key if you want to constrain the PDC pods to run only on the node you specify. For more information, see "nodeSelector" in the Kubernetes documentation. **Note**: To override the rules for a specific PDC pod, specify a different value for the pod's **nodeSelector** key. For example, set the **ocpdc.configEnv.transformation.syncPDC.nodeSelector** key to apply rules specifically to the SyncPDC pod.
affinity	**ocpdc**	The rules for running the following PDC pods on specific nodes: • pdc • RRE • BRE • SyncPDC • Import-Export Set this key if you want to constrain the PDC pod to run only on the nodes that meet your criteria. For more information about this key, see "Node Affinity" in the Kubernetes documentation. **Note**: To override the rules for a specific PDC pod, specify a different value for the pod's **affinity** key. For example, set the **ocpdc.configEnv.transformation.importExport.affinity** key to apply rules specifically to the Import-Export pod.
domainUID	**ocpdc.wop**	The name of the PDC WebLogic Server domain. The default is **pdc-domain**.
resources.*	**ocpdc**	The minimum and maximum CPU and memory resources for the PDC domain admin-server pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Table 7-5 (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
configEnv.*	**ocpdc**	The configuration details for PDC. • **name**: The name of the PDC ConfigMap. The default is **pdc-configmap-env**. • **upgrade**: Set to **true** to upgrade PDC. • **dbHostName**: The host name of the PDC and cross-reference database. The value must match that of **oc-cn-op-job-helm-chart**. • **dbPort**: The port for the PDC and cross-reference database. The value must match that of **oc-cn-op-job-helm-chart**. • **dbService**: The service name for the PDC and cross-reference database. The value must match that of **oc-cn-op-job-helm-chart**. • **dbSSLMode**: The type of SSL connection required for connecting to the cross-reference database: two-way SSL authentication is required (**TWO_WAY**), one-way SSL authentication is required (**ONE_WAY**), or SSL authentication is not required (**NO**). The default is **NO**.
extOPSSWallet **extPDCKeystoreSecret** **extPDCDBSSLWalletSecret**	**ocpdc.configEnv**	The names of the Kubernetes Secrets containing the custom SSL database wallet for PDC, the custom OPSS wallet for PDC, and custom KeyStore certificate file for PDC. This value must match the one for the **oc-cn-op-job-helm-chart**. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.

Table 7-5 (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
transformation.*	**ocpdc.configEnv**	The details for the transformation. • **logLevel**: The logging level, which can be **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, or **FINEST**. The default is **WARNING**. • **logFileSize**: The maximum file size, in bytes, of the log files. After the log file meets the maximum, PDC closes the log file and creates a new log file. The default is **500000**. See "Rotating PDC Log Files" in *BRM Cloud Native System Administrator's Guide* for more information. • **logFileCount**: The maximum number of log files to retain for the application. The default is **10**. • **persistTransactionLogs**: Whether to persist log files in the database after they are closed. Possible values are **all**, **disabled**, and **failed**. The default is **failed**, which specifies to persist only failed transaction logs. • **MEM_ARGS**: The memory argument, surrounded by quotes. For example: **"-Xms1024m -Xmx2048m -XX:CompileThreshold=8000"**. • **persistOutFiles**: Whether to keep the output and payload files generated by the transformation engine in the **out** directory (**enabled**) or remove the files from the out directory (**disabled**). The default is **disabled**. • **resources.***: The minimum and maximum CPU and memory resources for the RRE and BRE pods. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **nodeSelector**: The name of the node on which to run the RRE and BRE pods. • **affinity**: The rules for running the RRE and BRE pods on specific nodes.
seedData.*	**ocpdc.configEnv**	The details for deploying seed data. **Note**: If balance element and RUM data already exist in the PDC database, they are not overwritten. • **BE**: Whether to load sample balance elements into the PDC database when PDC is deployed (**true**) or not (**false**). The default is **false**. **Note**: If balance element data already exists in the PDC database, it is not overwritten. • **RUM**: Whether to load sample RUMs into the PDC database when PDC is deployed (**true**) or not (**false**). The default is **false**.

Table 7-5 (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
importExport.*	**ocpdc.configEnv**	The default for running the **ImportExportPricing** utility in PDC cloud native. For more information, see "Running PDC Applications" in *BRM Cloud Native System Administrator's Guide*. • **IE_Operation.***: The operation for the **ImportExportPricing** utility to perform. When you deploy PDC, ensure that this key has an empty value. – Empty value: No operation is performed. This is the default. – **export**: The utility exports data from the PDC database into an XML file. – **import**: The utility imports data from the XML file into the PDC database. – **publish**: The utility publishes components from PDC to the batch rating engine, real-time rating engine, or ECE. – **keep**: The utility retains the latest version of successfully promoted PDC components. – **deleteprofile**: The utility deletes pricing profiles from PDC. – **type**: The utility displays the pricing or setup components available in PDC. • **IE_Component.***: The type of components and objects to import or export using the **ImportExportPricing** utility. Don't include the hyphen (**-**) prefix with the value. – **config**: Imports or exports setup components, such as tax codes, business profiles, and general ledger IDs. – **pricing**: Imports or exports pricing components, such as events, charges, and chargeshares. – **metadata**: Imports or exports the event, service, account, and profile attribute specifications. – **profile**: Imports or exports pricing profile data. – **customfields**: Imports or exports custom fields. – **brmobject**: Exports BRM configuration objects such as services, events, and G/L IDs. – **all**: Imports or exports all objects and components. • **IE_File_OR_Dir_Name**: The name of the XML file or **ImportExport** directory that contains the list of components and objects to import into the PDC database. This XML file is used by the **ImportExportPricing** utility. If importing or deleting components, copy the XML file to the HostPath specified in **pdcBRMHostPath** or to pdc-brm-pvc. Set the file's or directory's ownership and permissions to **chown** *runAsUser***:0** and **chmod 777**. • **extraCmdLineArgs**: The extra command-line arguments for the **ImportExportPricing** utility, apart from operation, component, and file name. The value must be surrounded by quotes. For example, "**-n** *ObjectName***"**.

Table 7-5   (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
		• **logLevel**: Sets the logging level, which can be **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, or **FINEST**. The default is **WARNING**. • **logSize**: Sets the maximum file size, in bytes, of the log files. After the log file meets the maximum, PDC closes the log file and creates a new log file. • **logCount**: Specifies the maximum number of log files to retain for the application. See "Rotating PDC Log Files" in *BRM Cloud Native System Administrator's Guide* for more information. • **persistIELogs**: Specifies whether to persist log files in the database after they are closed. Possible values are **all**, **disabled**, and **failed**. The default is **failed**. • **resources.***: The minimum and maximum CPU and memory resources for the Import-Export pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **nodeSelector**: The rules for deploying the Import-Export pod on specific nodes. • **affinity**: The rules for deploying the Import-Export pod on specific nodes.

Table 7-5   (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
syncPDC.*	**ocpdc.configEnv**	The details for running the **SyncPDC** utility in PDC cloud native. For more information, see "Running PDC Applications" in *BRM Cloud Native System Administrator's Guide*. • **upgradeFromPS2**: Set this to true to upgrade from 12.0 Patch Set 2 to 12.0 Patch Set 4 or higher. • **logLevel**: Sets the logging level, which can be **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, or **FINEST**. The default is **WARNING**. • **logFileSize**: Sets the maximum file size, in bytes, of the log files. After the log file meets the maximum, PDC closes the log file and creates a new log file. The default is **20000**. • **logFileCount**: Specifies the maximum number of log files to retain for the application. The default is **10**. See "Rotating PDC Log Files" in *BRM Cloud Native System Administrator's Guide* for more information. • **MEM_ARGS**: The memory argument. • **skipBREMigration**: Skips the synchronization of pipeline configuration data. The default is **false**. This key is ignored when ECE is enabled in a PDC system. • **syncPDCStartAt**: The scheduled time for running the **SyncPDC** utility. This key is set at deployment time only. Valid values include: **startAt** and **"***HH***:***MM***"**. The default is **23:59**. • **syncPDCInterval**: The scheduled frequency for running the **SyncPDC** utility. This key is set at deployment time only. The default is **1:0**. • **enrichmentFileName**: Set this to **ECEEventEnrichmentSpec.xml**. Store the enrichment file in the path specified in **pdcBrmHostPath**. This is applicable at both PDC deployment time and individual **SyncPDC** runtime. • **runSyncPDC**: Whether to create the SyncPDC pod and start the BRM-to-PDC synchronization process (**true**) or delete the SyncPDC pod and stop the synchronization process (**false**). The default is **true**. • **resources.***: The minimum and maximum CPU and memory resources for the SyncPDC pod. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*. • **nodeSelector**: The rules for deploying the SyncPDC pod on specific nodes. • **affinity**: The rules for deploying the SyncPDC pod on specific nodes.
monitoring.*	**ocpdc.configEnv**	The details for monitoring PDC and collecting WebLogic metrics. By default, monitoring is disabled. For more information, see "Monitoring PDC in a Cloud Native Environment" in *BRM Cloud Native System Administrator's Guide*.

Table 7-5 (Cont.) Pricing Design Center Keys for oc-cn-helm-chart

Key	Path in values.yaml	Description
secretValue.*	**ocpdc**	The passwords for PDC. • **name**: The name of the Secret key. The default is **pdc-secret-env**. • **walletPassword**: The passwords for the PDC application wallet and PDC BRM integration wallet.

Adding PDC Keys for oc-cn-op-job-helm-chart

You must create an **override-values.yaml** for **oc-cn-op-job-helm-chart** and then add the PDC-specific keys in Table 7-6.

For information about all PDC-specific keys, see the descriptions in the **oc-cn-op-job-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocpdc.secretValue** hold sensitive data. Handle them carefully with controlled access to the file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 7-6 Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
isEnabled	**ocpdc**	Whether to enable PDC jobs: • **true**: Enables PDC jobs. This is the default. • **false**: Disables PDC jobs.
isClean	**ocpdc**	Whether to clean old PDC deployment and instance logs: • **true**: Removes all existing PDC deployment and instance logs. This is the default. • **false**: Keeps all existing PDC logs.
lang	**ocpdc**	The Linux system locale. The default is **en_US.UTF-8**.
tz	**ocpdc**	The Linux time zone. The default is **UTC**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
volume.domain.createOption.*	**ocpdc**	By default, PDC uses dynamic volume provisioning. To use a static volume instead, add the **createOption** keys under the **volume** section. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. **Note**: Ensure you provide the required permissions to the volume path by following the guidelines in "Persistent Volume Storage Locations" in the WebLogic Kubernetes Operator documentation.
storageSize	**ocpdc**	The storage size for the pdc-brm-pv or pdc-brm-pv, such as 10Gi.
enableSecurityContext	**ocpdc**	Whether to enable PDC pod-level security attributes and common container settings. The default is **true**.
deployment.*	**ocpdc**	The details for deploying the PDC image: • **imageName**: The name of the PDC image. The default is **oracle/pdc**. • **imageTag**: The tag name for the PDC image. The default is **:15.1.0.0.0**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
deployment.fmw.*	**ocpdc**	The details for pulling WebLogic images for PDC from a container registry: • **imageRepository**: The name of the container registry from which to pull the WebLogic image. The default is **container-registry.oracle.com/**. • **imageName**: The name of the container repository from which to pull the WebLogic image. The default is **middleware/fmw-infrastructure**. • **imageTag**: The tag name for the WebLogic image. See "Pulling WebLogic Images for PDC, Billing Care, Billing Care REST API, and Business Operations Center" for more information.
nodeSelector	**ocpdc**	The rules for scheduling a PDC job pod on a particular node using **nodeSelector** or **affinity**. **pdc-domain-job**: Set this key to constrain the PDC pods to run only on the node you specify. For more information, see "nodeSelector" in the Kubernetes documentation. **Note**: To override the rules for a specific PDC pod, specify a different value for the pod's **nodeSelector** key. For example, set the **ocpdc.configEnv.transformation.syncPDC.nodeSelector** key to apply rules specifically to the SyncPDC pod.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
affinity	**ocpdc**	The rules for scheduling a PDC job pod on a particular node using **nodeSelector** or **affinity**. **pdc-domain-job**: Set this key if you want to constrain the PDC pod to run only on nodes that meet your criteria. For more information, see "Node Affinity" in the Kubernetes documentation. **Note**: To override the rules for a specific PDC pod, specify a different value for the pod's affinity key. For example, set the **ocpdc.configEnv.transformation.importExport.affinity** key to apply rules specifically to the PDC Import Export pod.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
wop.*		The details for deploying the WebLogic Operator. • **domainUID**: The name of this PDC WebLogic Server domain. • **includeServerOutInPodLog**: Whether to include the server out file in the pod's stdout log (**true**) or not (**false**). The default is **true**. • **jtaTimeoutSeconds**: The maximum amount of time, in seconds, an active transaction is allowed to be in the first phase of a two-phase commit transaction. The default is **10000**. If the time expires, the transaction is automatically rolled back. • **jtaAbandonTimeoutSeconds**: The maximum amount of time, in seconds, a transaction manager continues to attempt completing the second phase of a two-phase commit transaction. The default is **10000**. • **stuckThreadMaxTime**: The number of seconds a thread must be continually working before the server considers the thread to be stuck. The default is **20000**. • **idlePeriodsUntilTimeout**: The number of idle periods until the peer is considered to be unreachable. The default is **40**. • **dataSourceXaTxnTimeout**: The number of seconds until the data source transaction times out. The default is **0**. When set to **0**, the WebLogic Server Transaction Manager passes the global WebLogic Server transaction timeout in seconds in the method. • **pdcAppSesTimeOut**: The PDC application (**pricingui.ear**) session time out, in seconds. The default is **36000**. • **pdcAppSesInvInterTimeOut**: The PDC application

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
		(**pricingui.ear**) session invalid interval time out, in seconds. The default is **3000**. • **maxMessageSize**: The maximum number of bytes allowed in messages that are received over supported protocols. The default is **10000000**. • **users.***: The users to add to the PDC domain. For more information, see "Creating PDC Users" in *BRM Cloud Native System Administrator's Guide*.
resources.*		The minimum and maximum CPU and memory resources for the PDC domain job. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
PDC Ports		The details about the PDC ports to use. • **exposePorts**: Exposes the SSL HTTPS port only (**yes**), the HTTP port only (**no**), or both (**all**). The default is **all**. • **t3ChannelPort**: The port number for the t3 channel. The default is **30799**. Use this key if PDC needs to use the t3 protocol to communicate with an external system, such as Elastic Charging Engine (ECE). Set this to a Kubernetes port number from 30000 through 32767 that is not in use. • **t3ChannelAddress**: The IP address for the primary node or load balancer. • **t3sChannelPort**: The port number for the t3s channel. The default is **30800**. Use this key if PDC needs to use the t3s protocol to communicate with an external system such as ECE. Set this to a Kubernetes port number from 30000 through 32767 that is not in use. • **t3sChannelAddress**: If SSL is enabled in the WebLogic domain, set this to the IP address for the primary node or load balancer.
Java Settings	**ocpdc.configEnv**	The Java security settings. • **USER_MEM_ARGS**: The custom memory arguments for WebLogic Admin Server. • **USER_JAVA_OPTIONS**: The custom Java options for WebLogic Admin Server. • **javaSecurityFileName**: The name of the Java security file listing the ciphers to disable. Place the file in the **pdc/java_security** directory.
tlsVersions	**ocpdc.configEnv**	The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
Log File Rotation	**ocpdc.configEnv**	The settings for rotating Pricing Server log and tracing log files: • **pdcAppLogLevel**: Sets the logging level, which can be **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, or **FINEST**. The default is **WARNING**. • **pdcAppLogFileSize**: Sets the maximum file size, in bytes, of the log files. After the log file meets the maximum, PDC closes the log file and creates a new log file. • **pdcAppLogFileCount**: Specifies the maximum number of log files to retain for the application. See "Rotating PDC Log Files" in *BRM Cloud Native System Administrator's Guide* for more information.
RCU Schema	**ocpdc.configEnv**	The details about the RCU schema. • **rcuJdbcURL**: The connection string for connecting to a database where schemas needed by Oracle Fusion Middleware products will be created, especially OPSS. Use the format "*host***:***port***/***service*". • **rcuPrefix**: The prefix for the PDC domain RCU schema. For example, if the prefix is **XYZ** and the schema name is **STB**, the PDC domain RCU schema name will be **XYZ_STB**. • **rcuRecreate**: Whether to re-create the PDC domain if present (**true**) or not (**false**).

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
isCustomWLSPython	**ocpdc.configEnv**	Whether to run your custom WebLogic Python files: • **true**: Run your custom WebLogic Python files located in **oc-cn-op-job-chart/pdc/customWLSPython**. • **false**: Do not run a custom WebLogic Python file. Set each file's ownership and permissions to **chown 1000:0** and **chmod 777**.
addOPSSWallet	**ocpdc.configEnv**	Whether you added the OPSS wallet to the Helm chart. • **true**: The OPSS wallet is added to the Helm chart. If set to **true**, you must copy the OPSS wallet file from **pdc-app-pvc/stores/opss_wallet/** to **oc-cn-op-job-helm-chart/pdc/opss_wallet**. • **false**: The OPSS wallet was not added to the Helm chart. This is the default. **Note**: The OPSS wallet file will not be available in **pdc-app-pvc/stores/opss_wallet** the first time you run **oc-cn-op-job-helm-chart** or create a new RCU prefix.
extOPSSWallet	**ocpdc.configEnv**	The name of the external Kubernetes Secret containing the custom OPSS wallet file for PDC. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
honorOMF	**ocpdc.configEnv**	Whether the RDS database honors the Oracle-Managed Files (OMF) naming format: • **true**: OMF format is used. • **false**: OMF format is not used. This is the default.
keyStoreType	**ocpdc.configEnv**	The SSL KeyStore type for the PDC domain. The default is **JKS**.
keyStoreAlias	**ocpdc.configEnv**	The alias name for the PDC domain SSL KeyStore. The default is **WeblogicPDCTestAlias**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
keyStoreIdentityFileName	**ocpdc.configEnv**	The name of the PDC domain SSL KeyStore Identity file. The default is **defaultserver.jks**. The **defaultserver.jks** file is created during PDC deployment if it does not already exist.
keyStoreTrustFileName	**ocpdc.configEnv**	The name of the PDC domain SSL TrustStore file. The default is **defaultclient.jks**. The **defaultclient.jks** file is created during PDC deployment if it does not already exist.
extPDCKeystoreSecret	**ocpdc.configEnv**	The name of the external Kubernetes Secret containing the custom KeyStore certificate file for PDC. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
isSSOEnabled	**ocpdc.configEnv**	Set to **true** to configure and use SAML 2.0 SSO service. The default is **false**.
extMetadataCM	**ocpdc.configEnv**	The name of the external ConfigMap containing the IDP metadata file.
samlAsserterName	**ocpdc.configEnv**	The name of the SAML Asserter. It should be the same as OEM or IDCS. The default is **pdcSAML2IdentityAsserter**.
ssoPublishedSiteURL	**ocpdc.configEnv**	The base URL used to construct endpoint URLs, typically, the load balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
ssoDefaultURL	**ocpdc.configEnv**	The URL to which unsolicited authentication responses are sent if they do not contain an accompanying target URL.
ssoLogoutURL	**ocpdc.configEnv**	The URL where users are redirected after they log out from the application (OEM or IDCS log out).

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
Database Details	**ocpdc.configEnv**	The details for • **dbHostName**: The host name of the PDC and cross-reference database. • **dbPort**: The port for the PDC and cross-reference database. • **dbService**: The service name for the PDC and cross-reference database. • **dbSysDBAUser**: The SYS, System, or Sys DBA user for the PDC and cross-reference database. If this key is not configured, PDC assumes that **pdcSchemaUserName** and **crossRefSchemaUserName** are already present in the database with the required permissions. • **dbSysDBARole**: The role of the PDC and cross-reference database SYS, System, or Sys DBA user. • **dbSSLMode**: The type of SSL connection required for connecting to the database: **TWO_WAY**, **ONE_WAY**, or **NO**. • **dbWalletType**: The type of file specified as the TrustStore for SSL connections: **SSO** or **pkcs12**. The default is **SSO**. If set to **ONE_WAY** or **TWO_WAY**, place the database wallet in the **oc-cn-helm-chart/pdc/pdc_db_wallet** directory. Create the directory structure if it is not present and do not change the directory name. • **extPDCDBSSLWalletSecret**: The name of the Kubernetes Secret containing the custom SSL database wallet file for PDC. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
Database Schema	**ocpdc.configEnv**	The details about the PDC database schema. • **crossRefSchemaPDCTableSpace**: The name of the PDC tablespace for the transformation cross-reference schema. This field is case-sensitive. • **crossRefSchemaTempTableSpace**: The name of the temporary tablespace for the transformation cross-reference schema. This field is case-sensitive. • **crossRefSchemaUserName**: The cross-reference database schema user name. • **pdcSchemaPDCTableSpace**: The tablespace name of the PDC schema. This field is case-sensitive. • **pdcSchemaTempTableSpace**: The tablespace name of the temporary schema. This field is case-sensitive. • **pdcSchemaUserName**: The PDC database schema user name. • **rcuWalletSchemaUserName**: The RCU wallet schema user name. The default schema user name is **PDCRCUWALLET**. **Note:** The OPSS wallet file created for the RCU schema is stored in the RCU_WALLET_DETAILS table during the first run. If the wallet file is available for the given RCU prefix, it is reused in subsequent runs and the RCU schema is not re-created. If the OPSS wallet file is present in **oc-cn-op-job-helm-chart/pdc/opss_wallet**, it takes precedence.
pdcAdminUser	**ocpdc.configEnv**	The PDC admin user name, which includes the Pricing Design Admin role. The default is **cnepdcadminuser**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
supportECE	**ocpdc.configEnv**	The charging engine to use: Elastic Charging Engine (**true**) or the real-time and batch rating engine (**false**). The default is **true**.
Upgrade	**ocpdc.configEnv**	The details for upgrading PDC. • **deployAndUpgradeSite2**: Set to **true** for zero-downtime upgrades (ZDU). The default is **false**. • **upgrade**: Set to **true** to upgrade from a previous version to a 15.*x* Patch Set or to deploy a 15.*x* interim patch. The default is **false**.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
secretValue.*	**ocpdc**	The credentials for accessing the system. • **adminPassword**: The password for the WebLogic domain's administrative user, which is used for accessing the WebLogic Console for administrative operations. • **rcuSchemaPassword**: The password for the Oracle Fusion Middleware product schemas that will be created by RCU and used by OPSS. • **keyStoreIdentityKeyPass**: The password for the PDC domain SSL Identity key. • **keyStoreIdentityStorePass**: The password for the PDC domain SSL identity store. • **keyStoreTrustStorePass**: The password for the PDC domain SSL TrustStore. • **dbPassword**: The Sys or System user password for the PDC and Cross Reference schema. • **pdcSchemaPassword**: The password for the PDC database schema user. • **crossRefSchemaPassword**: The password for the transformation cross-reference database schema user. • **rcuWalletSchemaPassword**: The password for the PDC RCU OPSS wallet schema. • **dbWalletPassword**: The password for the database SSL wallet. This key is required if **dbWalletType** is set to **pkcs12**. • **walletPassword**: The passwords for the PDC application wallet and PDC BRM integration wallet. • **pdcAdminUserPassword**: The password for the PDC admin user, which includes the Pricing Design Admin role.

Table 7-6 (Cont.) Pricing Design Center Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml	Description
service.*	**ocpdc**	The pdc-service service's details. • **name**: The name of the service: **pdc-service**. • **type**: The service type. The default is **ClusterIP**.

Setting Up SSO for PDC Cloud Native

SSO allows users to log in to applications using a single user name and password combination. You set up SSO for PDC cloud native services by using SAML 2.0.

To set up SSO for PDC:

1. Export the SAML 2.0 metadata XML file from your identity and access management (IAM) system.

   For example, if you are using Oracle Access Management, you can export the file by following the instructions in "Exporting Metadata" in *Oracle Fusion Middleware Administering Oracle Access Management*.

2. Add the metadata XML file to your BRM cloud native deployment by doing one of the following:

   - Rename the metadata XML file to **metadata.xml**, and then move **metadata.xml** to the **oc-cn-op-job-helm-chart/pdc/idp** directory.
   - Pre-create the IDP metadata ConfigMap for PDC and set the **ocpdc.configEnv.extMetadataCM** key in your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

3. Configure the KeyStores needed by SAML by doing one of the following:

   - Generate the Identity and Trust KeyStores and then move your files under the **oc-cn-op-job-helm-chart/pdc/pdc_keystore** directory.
   - Pre-create the Kubernetes Secret for the Identity and Trust KeyStore files and set the **ocpdc.configEnv.extKeystoreSecret** key in your **override-values.yaml** file for both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**.

     For more information, see "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.

4. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, set the following keys:

   - **ocpdc.configEnv.isSSOEnabled**: Set this to **true**.
   - **ocpdc.configEnv.keyStoreAlias**: Set this to the private key alias of the KeyStore.
   - **ocpdc.configEnv.keyStoreType**: Set this to the file type of the SSL Identity and Trust KeyStore. The only supported value is **JKS**.
   - **ocpdc.configEnv.keyStoreIdentityFileName**: Set this to the name of the Identity KeyStore file.
   - **ocpdc.configEnv.keyStoreTrustFileName**: Set this to the name of the Trust KeyStore file.
   - **ocpdc.configEnv.samlAsserterName**: Set this to the name of the SAML Asserter. The default is **pdcSAML2IdentityAsserter**.

- **ocpdc.configEnv.ssoPublishedSiteURL**: Set this to the base URL used to construct endpoint URLs. This is typically the load balancer host and port where the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
- **ocpdc.configEnv.ssoDefaultURL**: Set this to the URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
- **ocpdc.secretValue.keyStoreIdentityStorePass**: Set this to the StorePass for the Identity KeyStore.
- **ocpdc.secretValue.keyStoreIdentityKeyPass**: Set this to the KeyPass for the Identity KeyStore.
- **ocpdc.secretValue.keyStoreTrustStorePass**: Set this to the StorePass for the Trust KeyStore.

5. Configure your load balancer's rules to send responses to the PDC WebLogic domain with **/saml2** appended to the URL path.

   Note:

   Add this rule to your existing load balancer rules for routing responses to PDC (**/pdc**), the load balancer host name, and so on.

   See "Installing an Ingress Controller".
6. Deploy your PDC cloud native services by following the instructions in "Deploying BRM Cloud Native Services".
7. After PDC is deployed, retrieve the **sp-metadata-admin-server.xml** file from the **/shared/domains/***domainUID* directory in your container, where *domainUID* is the name of your PDC domain specified in the **ocpdc.wop.domainUID** key.

   The XML file configures the Web SSO Provider Partner. It contains the partner's KeyStore certificates, SAML assertion details, and the URLs where the SAML Identity Provider redirects to provide access to PDC.
8. Create a profile for your identity provider partner by loading the **sp-metadata-admin-server.xml** file into your IAM system.

   For example, if you are using Oracle Access Management, you can load the file by following the instructions in "Creating Remote Identity Provider Partners" in *Oracle Fusion Middleware Administering Oracle Access Management*.

Configuring Pipeline Configuration Center

To configure Pipeline Configuration Center (PCC) to run in your BRM cloud native environment:

1. Override the PCC-specific keys in the **values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding Pipeline Configuration Center Keys for oc-cn-op-job-helm-chart".
2. Override the PCC-specific keys in the **values.yaml** file for **oc-cn-helm-chart**. See "Adding Pipeline Configuration Center Keys for oc-cn-helm-chart".
3. Set up volume mounts. See "About PCC Volume Mounts".
4. Create a WebLogic domain and install the PCC application. See "Creating a WebLogic Domain and Installing the PCC Application".

5. Set up SAML for SSO in PCC. See "Setting Up SSO for Pipeline Configuration Center".
6. Set up local users and groups for PCC. See "Setting Up Local Users and Groups for PCC".
7. Start and stop your WebLogic servers. See "Starting and Stopping WebLogic Servers".
8. Enable SSL in PCC. See "Configuring SSL in PCC".

Adding Pipeline Configuration Center Keys for oc-cn-op-job-helm-chart

Table 7-7 lists the keys that directly impact PCC deployment. Add these keys to your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

Table 7-7 PCC Keys for oc-cn-op-job-helm-chart

Key	Paths in values.yaml File	Description
isEnabled	**ocpcc.pcc**	Whether to deploy, configure, and start PCC services: • **false**: Does not create the Kubernetes resources for using PCC. • **true**: Creates the Kubernetes resources for using PCC. This is the default.
imageName	**ocpcc.pcc.deployment.app**	The name of the PCC image, such as **oracle/pcc**
imageTag	**ocpcc.pcc.deployment.app**	The tag associated with the image. This is generally the patch set number prefixed with a colon (**:**). For example, :**15.1.0.0.0**
dbSSLMode	**ocpcc.pcc.configEnv**	The type of connection required to connect to the database: • **TWO_WAY**: Two-way SSL authentication is required. In this case, both the client and server must authenticate each others identity. • **ONE_WAY**: One-way SSL authentication is required. In this case, the client must authenticate the server's identity. This is the default. • **NO**: SSL authentication is not required.
dbWalletType	**ocpcc.pcc.configEnv**	The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO**or **PKCS12**.
rcuJdbcURL	**ocpcc.pcc.configEnv**	The connection string for connecting to the database where schemas needed by Oracle Fusion Middleware products will be created, especially OPSS.

Table 7-7 (Cont.) PCC Keys for oc-cn-op-job-helm-chart

Key	Paths in values.yaml File	Description
rcuDBARole	**ocpcc.pcc.configEnv**	The role of the database administrator user.
rcuArgs	**ocpcc.pcc.configEnv**	The additional arguments for creating the RCU.
ldapHost	**ocpcc.pcc.configEnv**	The host name or IP address of the LDAP Server (for example, OUD) where users and groups will be configured for access to PCC.
ldapPort	**ocpcc.pcc.configEnv**	The port number on which the LDAP server is listening.
ldapGroupBase	**ocpcc.pcc.configEnv**	The LDAP base DN that contains groups.
ldapUserBase	**ocpcc.pcc.configEnv**	The LDAP base DN that contains users.
extDBSSLWalletSecret **extKeystoreSecret**	**ocpcc.pcc.configEnv**	The names of the pre-created Kubernetes Secrets for the Pipeline Configuration Center KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
keystoreAlias	**ocpcc.pcc.configEnv**	The private key alias of the KeyStore.
keystoreType	**ocpcc.pcc.configEnv**	The file type of the SSL Identity and TrustStore, which is either **PKCS12**or **JKS**. The default is **PKCS12**.
keystoreIdentityFileName	**ocpcc.pcc.configEnv**	The file name of the Identity KeyStore.
keystoreTrustFileName	**ocpcc.pcc.configEnv**	The file name of the Trust KeyStore.
isSSOEnabled	**ocpcc.pcc.configEnv**	Whether to enable single sign-on (SSO) for PCC cloud native services through SAML 2.0: • **true**: **SSO** is enabled for PCC cloud native services. • **false**: **SSO** is disabled. This is the default.
samlAsserterName	**ocpcc.pcc.configEnv**	The name of the SAML Asserter. The default is **samlPCCAsserter**.

Table 7-7 (Cont.) PCC Keys for oc-cn-op-job-helm-chart

Key	Paths in values.yaml File	Description
ssoPublishedSiteURL	**ocpcc.pcc.configEnv**	The base URL that is used to construct endpoint URLs. This is typically the Load Balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
ssoDefaultURL	**ocpcc.pcc.configEnv**	The URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
reloadVersion	**ocpcc.pcc.configEnv**	Update this value with any value different from the current value to force a restart of the deployer.
adminPassword	**ocpcc.pcc.secretVal**	The password of the WebLogic domain's administrative user, which is used for accessing the WebLogic Console for administrative operations.
ldapPassword	**ocpcc.pcc.secretVal**	The password of the LDAP Server admin user.
rcuSysDBAPassword	**ocpcc.pcc.secretVal**	The password for the rcuJdbcURL database administrator.
rcuSchemaPassword	**ocpcc.pcc.secretVal**	The passwords for the schemas of Oracle Fusion Middleware products that will be created by RCU, which is used by OPSS.
dbWalletPassword	**ocpcc.pcc.secretVal**	The password for accessing the certificates from the TrustStore and KeyStore.
keystoreIdentityPassword	**ocpcc.pcc.secretVal**	The StorePass for the Identity KeyStore.
keystoreKeyPassword	**ocpcc.pcc.secretVal**	The KeyPass for the Identity KeyStore.
keystoreTrustPassword	**ocpcc.pcc.secretVal**	The StorePass for the Trust KeyStore.
domainUID	**ocpcc.pcc.wop**	The name of the domain. The default is **pcc-domain**.
adminChannelPort	**ocpcc.pcc.wop**	The NodePort where the admin-server's HTTP service is accessible.

Table 7-7 (Cont.) PCC Keys for oc-cn-op-job-helm-chart

Key	Paths in values.yaml File	Description
serverStartPolicy	**ocpcc.pcc.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count.
volume.*	**ocpcc.pcc**	Details about the PVC for the pcc pod: • **storage**: The storage size of the volume. • **createOption**: By default, the pcc pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocpcc.pcc**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocpcc.pcc**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.

Adding Pipeline Configuration Center Keys for oc-cn-helm-chart

Table 7-8 lists the keys that directly impact PCC deployment. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart**.

Table 7-8 Pipeline Configuration Center Keys

Key	Path in Values.yaml File	Description
appLogLevel	**ocpcc**	The logging level at which application logs must be captured in log files: **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**, and **ALL**.

Table 7-8 (Cont.) Pipeline Configuration Center Keys

Key	Path in Values.yaml File	Description
isEnabled	**ocpcc.pcc**	Whether to deploy, configure, and start PCC services: • **false**: Does not create the Kubernetes resources for using PCC. • **true**: Creates the Kubernetes resources for using PCC. This is the default.
imageName	**ocpcc.pcc.deployment.app**	The name of the PCC image, such as **oracle/pcc**.
imageTag	**ocpcc.pcc.deployment.app**	The tag associated with the image. This is generally the patch set number, prefixed with a colon (:). For example, :**15.1.0.0.0**
keystoreAlias	**ocpcc.pcc.configEnv**	The private key alias of the KeyStore.
dbSSLMode	**ocpcc.pcc.configEnv**	The type of connection required to connect to the database: • **TWO_WAY**: Two-way SSL authentication is required. In this case, both the client and server must authenticate each other’s identity. • **ONE_WAY**: One-way SSL authentication is required. In this case, the client must authenticate the server's identity. This is the default. • **NO**: SSL authentication is not required.
dbWalletType	**ocpcc.pcc.configEnv**	The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**.
extDBSSLWalletSecret **extKeystoreSecret**	**ocpcc.pcc.configEnv**	The names of the pre-created Kubernetes Secrets for the Pipeline Configuration Center KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator’s Guide*.
tlsVersions	**ocpcc.pcc.configEnv**	The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**.
login	**ocpcc.pcc.infranet.user**	The username of the service that has permission to access BRM.
serviceType	**ocpcc.pcc.infranet.user**	The POID type of the service that has permission to access BRM.
serviceID	**ocpcc.pcc.infranet.user**	The POID ID of the service that has permission to access BRM.
minSize	**ocpcc.pcc.infranet.connectionpool**	The minimum size of the connection pool.
maxSize	**ocpcc.pcc.infranet.connectionpool**	The maximum size of the connection pool.
loglevel	**ocpcc.pcc.infranet**	The log level for the infranet properties.
addOnProperties	**ocpcc.pcc.infranet**	Empty by default, you can use this key to specify custom infranet properties.
domainUID	**ocpcc.pcc.wop**	The name of the domain. The default is pcc-domain.

Table 7-8 (Cont.) Pipeline Configuration Center Keys

Key	Path in Values.yaml File	Description
adminChannelPort	**ocpcc.pcc.wop**	The NodePort where the admin-server's HTTP service will be accessible. The default is blank. **Note**: Set this key only if you want the **pcc-domain-admin-server-ext** service to deploy as NodePort.
serverStartPolicy	**ocpcc.pcc.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count.
isEnabled	**ocpcc.pcc.monitoring**	Whether to enable monitoring of PCC.
nodeSelector	**ocpcc.pcc**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocpcc.pcc**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.

Table 7-9 lists the secret keys that directly impact PCC deployment. These keys hold sensitive data and must be handled carefully with controlled access to the file containing its values. See "Secrets" in *Kubernetes Concepts*.

Add these secret keys to your **override-values.yaml** file, and encode all of their values in Base64.

Note:

- You can encode strings in Linux by using this command:

```
echo -n 'string' | base64
```

- You can decode strings in Linux by using this command:

```
echo 'encoded_string' | base64 --decode
```

Table 7-9 Pipeline Configuration Center Secret Keys

Key	Description
ocpcc.pcc.secretVal.adminPassword	The WebLogic Server administrative password encoded in Base64.
ocpcc.pcc.secretVal.walletPassword	The PCC wallet password encoded in Base64.
ocpcc.pcc.secretVal.rcuSysDBAPassword	The Database Administrator password encoded in Base64.

Table 7-9 (Cont.) Pipeline Configuration Center Secret Keys

Key	Description
ocpcc.pcc.secretVal.rcuSchemaPassword	The password for schemas of Oracle Fusion Middleware products that will be created by RCU, which is used by OPSS. The value must be Base64-encoded.
ocpcc.pcc.secretVal.keystoreIdentityPassword	The KeyPass of Identity Keystore, which is used for setting up the SSL-enabled domain. The value must be Base64-encoded.
ocpcc.pcc.secretVal.keystoreKeyPassword	The StorePass of Identity Keystore, which is used for setting up the SSL-enabled domain. This value must be Base64-encoded.
ocpcc.pcc.secretVal.keystoreTrustPassword	The StorePass of the Trust Keystore, which is used for setting up the SSL-enabled domain. This value must be Base64-encoded.
ocpcc.pcc.secretVal.pccUserPassword	The PCC user password encoded in Base64.

About PCC Volume Mounts

The PCC container requires Kubernetes volume mounts for sharing the domain and application file system between the WebLogic Cluster servers. There is one volume for the domain. By default, these are created dynamically, using the provisioner defined in BRM, in the **storage-class** key in **oc-cn-op-job-helm-chart**.

To change the volume type or provider, modify the following keys in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

- **ocpcc.pcc.volume.domain.createOption** for the domain file system for PCC.

Creating a WebLogic Domain and Installing the PCC Application

The WebLogic domain is created by a Kubernetes Deployment when **oc-cn-op-job-helm-chart** is installed. The same job also installs the PCC application and deploys the application EAR file onto the WebLogic Cluster.

The **oc-cn-op-job-helm-chart** chart also:

- Creates a Kubernetes ConfigMap and Secrets, which are used throughout the life-cycle of the WebLogic domain.
- Initializes the **PersistentVolumeClaim** for the domain and application file system as well as third-party libraries.

Note:

The **override-values.yaml** file that you use for this chart must include BRM override values.

After you install **oc-cn-op-job-helm-chart**, wait until the Kubernetes deployment has reached the **1/1 Running** status. Then, you can install or upgrade **oc-cn-helm-chart** for PCC services.

After the deployment is running, don't delete the chart. Its resources will be used for starting and stopping the servers through **oc-cn-helm-chart**.

Setting Up SSO for Pipeline Configuration Center

SSO allows users to log in to applications using a single user name and password combination. You set up SSO for Pipeline Configuration Center cloud native services by using SAML 2.0.

To set up SSO for Pipeline Configuration Center:

1. Export the SAML 2.0 metadata XML file from your identity and access management (IAM) system.

   For example, if you are using Oracle Access Management, you can export the file by following the instructions in "Exporting Metadata" in *Oracle Fusion Middleware Administering Oracle Access Management*.

2. Add the metadata XML file to your BRM cloud native deployment by doing one of the following:

   - Rename the metadata XML file to **metadata.xml**, and then move **metadata.xml** to the **oc-cn-op-job-helm-chart/pcc/idp** directory.
   - Pre-create the IDP metadata ConfigMap for Pipeline Configuration Center and set the **ocpcc.pcc.configEnv.extMetadataCM** key in your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

3. Configure the KeyStores needed by SAML by doing one of the following:

   - Generate the Identity and Trust KeyStores and then move your files, such as **identity.p12** and **trust.p12**, under the **oc-cn-op-job-helm-chart/pcc/keystore** directory.
   - Pre-create the Kubernetes Secret for the Identity and Trust KeyStores and set the **ocpcc.pcc.configEnv.extKeystoreSecret** key in your **override-values.yaml** file for both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

4. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **isSSOEnabled** key to **true**.

5. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, set the following keys:

   - **ocpcc.pcc.configEnv.isSSOEnabled**: Set this to **true**.
   - **ocpcc.pcc.configEnv.keystoreAlias**: Set this to the private key alias of the KeyStore.
   - **ocpcc.pcc.configEnv.keystoreType**: Set this to the file type of the SSL Identity and Trust store, which is either **PKCS12** or **JKS**. The default is **PKCS12**.
   - **ocpcc.pcc.configEnv.keystoreIdentityFileName**: Set this to the name of the Identity KeyStore file.
   - **ocpcc.pcc.configEnv.keystoreTrustFileName**: Set this to the name of the Trust KeyStore file.
   - **ocpcc.pcc.configEnv.samlAsserterName**: Set this to the name of the SAML Asserter. The default is **samlPCCAsserter**.

- **ocpcc.pcc.configEnv.ssoPublishedSiteURL**: Set this to the base URL that is used to construct endpoint URLs. This is typically the load balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
- **ocpcc.pcc.configEnv.ssoDefaultURL**: Set this to the URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
- **ocpcc.pcc.secretVal.keystoreIdentityPassword**: Set this to the StorePass for the Identity KeyStore.
- **ocpcc.pcc.secretVal.keystoreKeyPassword**: Set this to the KeyPass for the Identity KeyStore.
- **ocpcc.pcc.secretVal.keystoreTrustPassword**: Set this to the StorePass for the Trust KeyStore.

6. Configure your load balancer's rules to send responses to the Pipeline Configuration Center WebLogic domain with **/saml2** appended to the URL path.

   Note:

   Add this rule to your existing load balancer rules for routing responses to Pipeline Configuration Center (**/pcc**), the load balancer host name, and so on.

   See "Installing an Ingress Controller".

7. Deploy your Pipeline Configuration Center cloud native services by following the instructions in "Deploying BRM Cloud Native Services".

8. After Pipeline Configuration Center is deployed, retrieve the **sp-metadata-admin-server.xml** file from the **/shared/domains/***domainUID* directory in your container, where *domainUID* is the name of your Pipeline Configuration Center domain specified in the **ocpcc.pcc.wop.domainUID** key.

   The XML file configures the Web SSO Provider Partner. It contains the partner's KeyStore certificates, SAML assertion details, and the URLs where the SAML Identity Provider redirects to provide access to Pipeline Configuration Center.

9. Create a profile for your identity provider partner by loading the **sp-metadata-admin-server.xml** file into your IAM system.

   For example, if you are using Oracle Access Management, you can load the file by following the instructions in "Creating Remote Identity Provider Partners" in *Oracle Fusion Middleware Administering Oracle Access Management*.

Setting Up Local Users and Groups for PCC

You have the option to customize the values for **oc-cn-op-job-helm-chart** to create users and groups locally in Oracle WebLogic Server. This would be especially useful for test environments where you might not have Identity Providers or LDAPs available. The groups for the admin user for WebLogic Server cannot be modified using this procedure.

Any passwords must be encoded using Base64. You can leave the password blank, but then the user will not be able to log in to the application directly.

To set up local users and groups for PCC, define the keys under **ocpcc.pcc.wlsUserGroups** in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

A group has to be the Config Admin to access the PCC UI. Only a user associated with the Config Admin group has full access to the PCC user interface. For example:

```
Add users and groups to domain's DefaultAuthenticator (local)
        wlsUserGroups:
            groups:
            - name: Config Admin
              description: PCC Admin
            # Each element for this takes "name", "description", "password" (base64
encoded) and list of "groups" that he is part of, like:
            # - name:
            # description:
            # password:
            # groups:
            # - "Regular CSR"
            users:
            - name: pccuser
              description: pccuser
              password: QzFnMmIzdTQj
              groups:
              - "Config Admin"
```

Starting and Stopping WebLogic Servers

When you install **oc-cn-op-job-helm-chart**, the default configuration sets up a WebLogic Cluster with five Managed Servers. When you install or upgrade **oc-cn-helm-chart** for the PCC service, two of the Managed Servers and one Admin Server are started.

By modifying the **override-values.yaml** file for **oc-cn-helm-chart**, you can control:

- The total number of Managed Servers and the initial server start up by using the **totalManagedServers** and **initialServerCount** keys.
- Whether the servers are started or stopped by using the **serverStartPolicy** key. To start the Admin Servers and the Managed Servers in a Cluster, set the key to **IF_NEEDED**. To stop all servers, set the key to **NEVER**.

Note:

The keys in the **override-values.yaml** file should be the same as the ones used in **oc-cn-op-job-helm-chart** for keys that are common in both charts.

After you modify the **override-values.yaml** file, update the Helm release for the changes to take effect:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

Configuring SSL in PCC

To access PCC over the HTTPS port, SSL must be enabled in the WebLogic domain where PCC is deployed. The BRM cloud native deployment package takes care of the configuration necessary to equip the WebLogic domain with SSL access.

To complete the configuration for SSL setup:

1. Copy PKCS12 files with valid certificates to the **oc-cn-op-job-helm-chart/pcc/keystore** directory:

   - **identity.p12:** Provides the certificate to identify the server.
   - **trust.p12:** Establishes trust for the certificate.

   If your KeyStore files have different file names or file types, such as JKS, override the **keyStoreIdentityFileName**, **keyStoreTrustFileName**, and **keyStoreType** keys in the **override-values.yaml** file for **oc-cn-helm-chart**.

   The **keystoreAlias** key is also mandatory along with **keyStoreIdentityFileName**, **keystoreTrustFileName** to enable SSL.

8

Configuring REST Services

Learn how to integrate external applications with your Oracle Communications Billing and Revenue Management (BRM) cloud native environment by using the BRM and PDC REST services.

Topics in this document:

- Configuring BRM REST Services Manager
- Configuring PDC REST Services Manager

Configuring BRM REST Services Manager

You use BRM REST Services Manager to integrate an external customer experience application with BRM. This allows you to manage billing and rating in BRM and then view your customers' account balances and bills in your external client. For more information, see *REST Services Manager API for Billing and Revenue Management*.

To configure BRM REST Services Manager in BRM cloud native:

1. Generate an SSL certificate. See "Generating an SSL Certificate for BRM REST Services Manager".
2. Optionally, configure the BRM REST Services Manager SDK. See "Configuring the SDK (Optional)".
3. If BRM and REST Services Manager are located in separate clusters, connect BRM REST Services Manager to BRM. See "Connecting to a Separate BRM Cluster".
4. Override the BRM REST Services Manager-specific keys in the **values.yaml** file. See "Adding BRM REST Services Manager Keys".
5. Optionally, configure policies for authorizing calls to the BRM REST Services Manager REST API endpoints. See "Configuring Policies for API Authorization".

Generating an SSL Certificate for BRM REST Services Manager

The following shows the steps for generating a sample SSL certificate:

1. Create a directory named **rsm_keystore** under the **oc-cn-helm-chart/rsm** directory.
2. Generate an SSL certificate. For example:

   ```
   openssl req -x509 -newkey rsa:4096 -keyout openSSLKey.pem -out cert.pem -days 365 -
   nodes
   ```

3. Generate a **PKCS12** KeyStore file. For example, this creates a KeyStore file named **keystore.p12**:

   ```
   openssl pkcs12 -export -out keyStore.p12 -inkey openSSLKey.pem -in cert.pem
   ```

4. Copy your SSL certificate file to the **oc-cn-helm-chart/rsm/rsm_keystore** directory.

Configuring the SDK (Optional)

To integrate the SDK with BRM REST Services Manager, generate an SDK image as follows:

1. Copy your extended SDK JAR **oc-cn-docker-files-15.1.***x.x.x***/oc-cn-docker-files/ocrsm/brm_rest_services_manager/SDK/libs** to the **oc-cn-docker-files-15.1.***x.x.x***/oc-cn-docker-files/ocrsm/brm_rest_services_manager/SDK** directory.

   Note:

   The SDK JAR can be used directly from **oc-cn-docker-files-15.1.***x.x.x***/oc-cn-docker-files/ocrsm/brm_rest_services_manager/SDK/libs** if no changes are required. If you need to make further customizations, follow the instructions in *REST Services Manager API for Billing and Revenue Management* and then copy the updated SDK JAR to the **oc-cn-docker-files-15.1.***x.x.x***/oc-cn-docker-files/ocrsm/brm_rest_services_manager/SDK** directory.

2. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **ocrsm.rsm.configEnv.rsmExtensionJar** key to the name of your extended SDK JAR file, such as **BRMRESTExtension.jar**.
3. Go to the **oc-cn-docker-files-15.1.***x.x.x***/oc-cn-docker-files/ocrsm/brm_rest_services_manager/SDK** directory.
4. Build the Podman image by running this command:

   ```
   podman build --format docker --tag imagerepo/brm-rest-services-manager-extension:1 .
   ```

5. Push the SDK image to the repository by running this command:

   ```
   podman login --username user --password password imagerepo
   podman push imagerepo/brm-rest-services-manager-extension:1
   ```

Connecting to a Separate BRM Cluster

If BRM is located in a separate cluster from BRM REST Services Manager, do the following to connect BRM REST Services Manager to BRM:

1. Open the **configmap_env_brmrsm.yaml** file in a text editor.
2. Set **BRM_HOST_NAME** to the host name of the cluster on which BRM is located. The default value is **cm**.
3. Save and close the file.

Adding BRM REST Services Manager Keys

Table 8-1 lists the keys that directly impact BRM REST Services Manager. Add these keys to your **override-values.yaml** file with the same path hierarchy.

Caution:

Keys with the path **ocrsm.rsm.secretVal** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64. See "Secrets" in *Kubernetes Concepts*.

Table 8-1 BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
isEnabled	**ocrsm.rsm**	Whether to deploy BRM REST Services Manager with BRM cloud native (true) or not (false). The default is **true**.
labels.*	**ocrsm.rsm**	The string used to form the names of BRM REST Services Manager.
deployment.*	**ocrsm.rsm**	The details for deploying BRM REST Services Manager. • **deadlineSeconds**: The maximum time, in seconds, for a deployment to make progress before it is considered failed. The default is **120**. • **revisionHistLimit**: The maximum number of old ReplicaSets for this deployment to retain. The remaining is garbage-collected in the background. The default is **10**. • **imageName**: The name of the BRM REST Services Manager image, such as **oracle/brm-rest-services-manager**. • **imageTag**: The tag associated with the image, such as **:15.1.0.0.0**.
sdk.*	**ocrsm.rsm.deployment**	The details about the BRM REST Services Manager SDK. • **imageName**: The name of the BRM REST Services Manager SDK image, such as **brm-rest-services-manager-extension**. • **imageTag**: The tag associated with the BRM Services Manager SDK image, such as **:1**.
probe.ready.*	**ocrsm.rsm.deployment**	The configuration for the readiness probe. • **delayInSec**: The duration, in seconds, to wait before performing the first readiness probe. The default is **30**. • **intervalInSec**: How often, in seconds, to perform the readiness probe. The default is **5**. • **maxAttempts**: The maximum number of consecutive failures before the probe is considered failed. The default is **15**.

Table 8-1 (Cont.) BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
REST Services Manager Volumes	**ocrsm.rsm.deployment**	The details about the REST Services Manager volumes. • **volMntKeyStore.name**: The volume containing the BRM REST Services Manager SSL KeyStore certificate. • **volMntLogs.name**: The name of the log volume mount where the SDK JAR is mounted. • **volMntSDK.name**: The name of the SDK volume mount where the SDK Jar is mounted. • **volMntSecretEnv.name**: The name of the volume mount that holds all passwords as a Secret. • **volMntSecurity.***: The name and path of the volume mount containing the server security. • **volMntAppExternalProperties.***: The name and path of the volume mount containing the application's external properties.
configEnv.*	**ocrsm.rsm**	The configuration details for the BRM REST Services Manager API. • **name**: The name of the BRM REST Services Manager API ConfigMap. The default is **brm-rest-services-manager-env-configmap**. • **httpPort**: The HTTP port in the container on which to deploy BRM REST Services Manager. The default is **9090**. • **httpsPort**: The HTTPS port in the container on which to deploy BRM REST Services Manager. The default is **8080**. • **adminPort**: The administration port for health, metrics, and other administration-related activities. The default is **9060**. • **tlsVersions**: The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**. • **rsmCertificateFileName**: The SSL certificate file name for BRM REST Services Manager. • **trustStoreFileName**: This is the optional file name for the TrustStore. Set this key if the default Java TrustStore needs to be overridden.

Table 8-1 (Cont.) BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
infranet.*	**ocrsm.rsm.config Env**	The details for connecting to BRM cloud native. • **user.login**: The login name of the service with permissions to access BRM. • **user.serviceType**: The POID type for the service having permissions to access BRM. • **user.serviceId**: The POID of the service having permissions to access BRM. • **connectionpool.minSize**: The minimum number of threads in the connection pool. • **connectionpool.maxSize**: The maximum number of threads in the connection pool. • **PcmTimeoutInMsecs**: The duration, in milliseconds, the PCM waits before timing out. • **login.type**: Whether a login name and password are required to connect to the BRM database (**1**) or not (**0**). The default is **1**.
brmSSLWalletFile Name	**ocrsm.rsm.config Env**	The BRM SSL wallet file name. The default is **cwallet.sso**.
rsmExtensionJar	**ocrsm.rsm.config Env**	The file name of the BRM REST Service Manager SDK JAR, such as **BRMRESTExtension.jar**.
bipURL	**ocrsm.rsm.config Env**	The Oracle Analytics Publisher URL.
bipUserId	**ocrsm.rsm.config Env**	The Oracle Analytics Publisher user ID.
Zipkin Tracing Details	**ocrsm.rsm.config Env**	The Zipkin tracing details. • **isTracingEnabled**: Whether to enable tracing for the BRM REST Services Manager API (**true**) or not (**false**). The default is **false**. • **zipkinHostName**: The host name of the Zipkin tracing collector. • **zipkinPort**: The port of the Zipkin tracing collector. • **zipkinProtocol**: The protocol of the Zipkin tracing collector, such as **http** or **https**.

Table 8-1 (Cont.) BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
BRM REST Services Manager Cache Details	**ocrsm.rsm.configEnv**	The cache details. • **baseURL**: The base URL with resource details to return in the response of BRM REST Services Manager requests. **Note:** After deployment, you can update this value by editing your **override-values.yaml** file and then doing a Helm upgrade. • **cacheEnabled**: Whether BRM REST Services Manager supports an internal cache (**true**) or not (**false**). The default is **false**. • **cacheType**: The cache type: **private** or **public**. The default is **private**. • **cacheMaxAge**: The time, in seconds, that the response remains fresh in cache after the response is generated. The default is **86400**. • **cacheServices**: The services that need to provide cache support. The default is **describe**, which is the resource locator for the Describe API in the REST API path.
securityEnabled	**ocrsm.rsm.configEnv**	The flag to indicate if token-based authentication is enabled for BRM REST Services Manager (**true**) or not (**false**). The default is **true**.
oidc.*	**ocrsm.rsm.configEnv**	The Identity Provider (IdP) authentication details. • **identity-uri**: The URI of the Identity Server, used as the base URL to retrieve metadata from the Identity Server. • **client-id**: The client ID generated by the Identity Server, used to validate the token. • **client-secret**: The client secret generated by the Identity Server, used to authenticate the application when requesting a JWT based on a code. Do not set this key directly. Instead, specify the client secret password under the **secretVal.clientSecret** key. • **proxyHost**: The proxy host of the IdP, if defined. When set, this triggers the use of a proxy for HTTP requests. • **scope-audience**: The audience for the scope required by this application. This is prefixed to the scope name when requesting scopes from the Identity Server. • **audience**: The secondary audience configured in the IdP. If no secondary audience is configured, use the primary audience, which is the same as the scope-audience. • **introspect-endpoint-uri**: The endpoint URI used to validate the JWT.
roleMapperName	**ocrsm.rsm.configEnv**	To authorize requests, specify a mapper based on your Identity Provider (IdP): use **idcs-role-mapper** for Oracle IDCS or **oam-role-mapper** for Oracle Access Manager. This step is necessary when your JSON Web Tokens (JWTs) do not adhere to the MicroProfile JWT RBAC v2.1 specification and lack a "groups" claim because it enables fetching user/client groups and roles. If your JWTs conform to the JWT RBAC v2.1 specification, leave this key empty.

Table 8-1 (Cont.) BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
oam.*	**ocrsm.rsm.config Env**	The Oracle Access Manager authentication details. These keys ensure backward compatibility for users who have not yet migrated to the latest versions of Oracle Access Management, where roles and groups are included in the JWT token. **Note**: This section is enabled only if **roleMapperName** is set to **oam-role-mapper**. • **oudHostName**: The Oracle Unified Directory host name. • **oudRootUserDN**: The Oracle Unified Directory root user domain name. • **oudHttpPort**: The Oracle Unified Directory HTTP port. • **oudHttpsPort**: The Oracle Unified Directory HTTPS port. • **oudUserBaseDN**: The Oracle Unified Directory user domain name. • **oudGroupDN**: The Oracle Unified Directory group domain name. • **msgType**: The message type based on the schema used to search roles in the Oracle Unified Directory. • **filter**: The filter based on the user attribute.
Log Level Details	**ocrsm.rsm.config Env**	The logging level. The possible values for these keys are: **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**. • **logLevel**: The application log level. • **helidonSecurityLogLevel**: The Helidon security log level. • **helidonWebServerLogLevel**: The Helidon WebLogic server log level. • **helidonConfigLogLevel**: The Helidon configuration log level. • **helidonMicroProfileLogLevel**: The Helidon MP log level. • **helidonCommonLogLevel**: The Helidon common log level. • **nettyServerLogLevel**: The embedded netty server log level. • **jerseyLogLevel**: The Jersey log level. • **jbossWeldLogLevel**: The Helidon JBossWeld log level. • **auditLogLevel**: The audit log level.
jvmOpts	**ocrsm.rsm.config Env**	The Java options to configure when setting resources for the containers, such as heap memory. For example: -XX:InitialRAMPercentage=25.0 -XX:MaxRAMPercentage=50.0.

Table 8-1 (Cont.) BRM REST Services Manager Keys

Key	Path in Values.yaml File	Description
extKeystoreSecret	**ocrsm.rsm.secretKeyStore**	The name of the external Kubernetes Secret containing the Identity and Trust files. **Note:** Override the **rsmCertificateFileName**, **trustStoreFileName**, and **keyStoreFileName** with the respective names of the TrustStore and KeyStore. See "Secrets" in *Kubernetes Concepts*, and "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
secretVal.*	**ocrsm.rsm**	The credentials for accessing the system. • **name**: The name of the Kubernetes Secret that copies certificates to the container. The default is **brm-rest-services-manger-env-secret**. • **rsmCertificatePassword**: The Base64-encoded certificate password for BRM REST Services Manager. • **brmInfranetWalletPassword**: The Base64-encoded wallet password. You can use any password. This password will be used to store the Oracle Analytics Publisher and Infranet connections in the wallet and can be used to access the same. • **bipPassword**: The Base64-encoded Oracle Analytics Publisher password. • **clientSecret**: The Base64-encoded IDCS client secret. • **trustStorePassword**: The TrustStore file password.
affinity	**ocrsm.rsm**	The rules for scheduling pods on particular nodes using more powerful selectors using affinity rules.
resources.*	**ocrsm.rsm**	The minimum and maximum CPU and memory resources that containers can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
hpa.*	**ocrsm.rsm**	The details for scaling up or down the number of pod replicas in your deployment based on a pod's CPU or memory utilization. By default, the Horizontal Pod Autoscaler is disabled. See "Setting Up Autoscaling of BRM Pods" in *BRM Cloud Native System Administrator's Guide*.
service.*	**ocrsm.rsm**	The brm-rest-services-manager service's details. • **name**: The name of the service: **brm-rest-services-manager**. • **type**: The service type. The default is **ClusterIP**.

You can use the following commands to encode and decode passwords in Base64 format:

- To encode strings in Linux:

  ```
  echo -n 'password' | base64
  ```

- To decode strings in Linux:

  ```
  echo 'encoded_password' | base64 --decode
  ```

Configuring Policies for API Authorization

To configure the policies for API authorization:

1. Define the API authorization rules in a policy file.

   You can use the sample authorization policy ConfigMap (**configmap_auth_policy_brmrsm.yaml**) as a template for defining API authorization rules.

2. For any new BRM REST Services Manager API endpoints, ensure that appropriate policy statements are added to the file. This is essential for enforcing proper authorization and access restrictions for each new API.

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

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

Configuring PDC REST Services Manager

You use PDC REST Services Manager to integrate an enterprise product catalog, such as Oracle Digital Experience for Communications Launch Experience, with PDC. This enables you to create a variety of product offerings in your enterprise product catalog and then have all of the rating and billing performed by PDC and BRM. For more information, see "About PDC REST Services Manager" in *PDC REST Services Manager Integration Guide*.

To configure PDC REST Services Manager in BRM cloud native:

1. Override the PDC REST Services Manager-specific keys in the **values.yaml** file. See "Adding PDC REST Services Manager Keys".

2. Configure OAuth authentication:

   a. If you are using Oracle Access Management for OAuth, create an identity domain, resource server, and OAuth client for PDC REST Services Manager in Oracle Access Management as described in "Setting Up OAuth for PDC REST Services Manager with Oracle Access Management" in *BRM Security Guide*.

   b. Configure the keys in the **override-values.yaml** file for OAuth with either Oracle Identity Cloud Service or Oracle Access Management as described in "Configuring OAuth Authentication in PDC REST Services Manager".

3. Configure outbound communication to the enterprise product catalog. See "Configuring Requests to the Enterprise Product Catalog".

4. Enable TLS encryption in PDC REST Services Manager to secure the communications it receives from your enterprise product catalog. See "Enabling TLS in PDC REST Services Manager".

5. Enable the T3S protocol in PDC REST Services Manager to secure its communications to PDC. See "Enabling T3S in PDC REST Services Manager".

6. Map TMF620 priceType values to BRM events to ensure that PDC REST Services Manager triggers the correct charging events for your pricing components. See "Configuring Mapping of TMF620 priceType to BRM Events".

Adding PDC REST Services Manager Keys

Table 8-2 lists the keys that directly impact PDC REST Services Manager. Add these keys to your **override-values.yaml** file with the same path hierarchy.

Caution:

Keys with the path **ocpdcrsm.secretValue** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64. See "Secrets" in *Kubernetes Concepts*.

Table 8-2 PDC REST Services Manager Keys

Key	Path in Values.yaml File	Description
isEnabled	**ocpdcrsm**	Whether to enable and deploy PDC REST Services Manager with BRM cloud native (**true**) or not (**false**). The default is **false**.
labels.name	**ocpdcrsm**	The string used to form the names of PDC REST Services Manager. The default is **pdcrsm**.
deployment.*	**ocpdcrsm**	The details for deploying PDC REST Services Manager. • **deadlineSeconds**: The maximum time, in seconds, for a deployment to make progress before it is considered failed. The default is **60**. • **revisionHistLimit**: The maximum number of old ReplicaSets for this deployment to retain. The remaining will be garbage-collected in the background. The default is **10**. • **imageName**: The name of the PDC REST Services Manager image, such as **oracle/pdcrsm**. • **imageTag**: The tag associated with the image, such as **:15.1.0.0.0**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**. • **rootLogLevel**: The root log level. The default is **INFO**. • **appLogLevel**: The application log level. The default is **INFO**. • **JAVA_OPTS**: The Java options to configure. • **JAVA_MEM_OPTS**: The Java memory options to configure.
resources.*	**ocpdcrsm.deployment**	The minimum and maximum CPU and memory resources that containers can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Table 8-2 (Cont.) PDC REST Services Manager Keys

Key	Path in Values.yaml File	Description
configEnv.*	**ocpdcrsm**	The configuration details for the PDC REST Services Manager. • **name**: The name of the PDC REST Services Manager API ConfigMap. The default is **pdcrsm-configmap-env**. • **rsmListenerPort**: The HTTPS port number assigned to listen for API requests from the enterprise product catalog. The default is **8080**. • **baseUrl**: The base URL with resource details to return in the response of PDC REST Services Manager requests. **Note:** After deployment, you can update this value by editing your **override-values.yaml** file and then doing a Helm upgrade. • **useT3s**: Whether to use T3s for the connection to PDC (**true**) or not (**false**). The default is **true**. • **securityEnabled**: Whether to enable token-based authentication for PDC REST Services Manager (**true**) or not (**false**). • **securityType**: Which OAuth provider to use for token-based authentication. Set this to **oam** for Oracle Access Management or **idcs** for Oracle Identity Cloud Service.
idcs.*	**ocpdcrsm.configEnv**	The Oracle Identity Cloud Service (IDCS) authentication details. • **inboundOauthUri**: The PDC REST Services Manager inbound OAuth base URI. • **inboundOauthClientId**: The PDC REST Services Manager client ID for inbound OAuth. • **inboundOauthFrontendUri**: The front end URI for inbound OAuth. • **inboundOauthAudience**: The primary audience for inbound OAuth. • **inboundOauthProxyHost**: The proxy host for inbound OAuth, if required. • **inboundOauthPubEventScope**: The scope required to access the TMF 620 Publish Event endpoint for inbound OAuth. • **inboundOauthMetricsScope**: The scope required to access the metrics endpoint for inbound OAuth.

Table 8-2 (Cont.) PDC REST Services Manager Keys

Key	Path in Values.yaml File	Description
oam.*	**ocpdcrsm.config Env**	The Oracle Access Manager authentication details. • **domainName**: The Oracle Access Manager domain name. • **audience**: The name of the Oracle Access Manager OAuth server. • **endpointURL**: The OAuth token endpoint for Oracle Access Manager. • **introspectendpointuri**: The introspect endpoint for Oracle Access Manager. • **scopeaudience**: The OAuth scope audience for Oracle Access Manager. • **authorizationendpointuri**: The OAuth authorization endpoint for Oracle Access Manager. • **proxyhost**: The OAuth proxy host for Oracle Access Manager. • **frontenduri**: The OAuth front end URI for Oracle Access Manager.
isTracingEnabled	**ocpdcrsm.config Env**	Whether to enable tracing for the PDC REST Services Manager API (**true**) or not (**false**). The default is **false**.
isTlsEnabled	**ocpdcrsm.config Env**	Whether to enable TLS encryption for PDC REST Services Manager (**true**) or not (**false**). The default is **false**.
tlsVersions	**ocpdcrsm.config Env**	The list of supported TLS versions, such as **TLSv1.2,TLSv1.3**.
tlsCertificateFile	**ocpdcrsm.config Env**	The path to the TLS certificate bundle relative to the Helm chart. The certificate must be in PKCS12 format. Ensure that the certificate file in the **rsm** directory.
extPDCRSMTlsSecret	**ocpdcrsm.config Env**	The name of the pre-created Kubernetes Secret for the custom TLS certificate. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
httpClients.*	**ocpdcrsm.config Env**	The details for configuring the HTTP client.
monitoring.prometheus*	**ocpdcrsm.config Env**	The details for monitoring PDC REST Services Manager. • **isEnabled**: Whether you are using Prometheus Operator to monitor PDC REST Services Manager (**true**) or not (**false**). The default is **false**. • **namespace**: The namespace in which to deploy Prometheus Operator. The default is **monitoring**.
secretVal.*	**ocrsm.rsm**	The credentials for accessing the system. • **name**: The name of the Kubernetes Secret that copies certificates to the container. The default is **pdcrsm-secret-env**. • **inboundOauthClientSecret**: The client secret for PDC REST Services Manager inbound OAuth. • **tlsCertificatePassphrase**: The Base64-encrypted passphrase for the TLS certificate. • **httpClients.***: The HTTP client configuration.

Table 8-2 (Cont.) PDC REST Services Manager Keys

Key	Path in Values.yaml File	Description
service.*	**ocpdcrsm**	Details about the pdcrsm service: • **name**: The service name: **pdcrsm**. • **type**: The service type. The default is **ClusterIP**. • **nodePort**: The external node port. The default is **31000**. **Note**: The **nodePort** key is applicable only when the **type** key is set to **NodePort**.

Sample PDC REST Services Manager override-values.yaml Entries

The following shows sample content in the **override-values.yaml** for PDC REST Services Manager, when Oracle Access Management is used for OAuth authentication:

```
ocpdcrsm:
    isEnabled: true
    labels:
        name: "pdcrsm"
    deployment:
        deadlineSeconds: 60
        revisionHistLimit: 10
        imageName: "oracle/pdcrsm"
        imageTag: ":15.1.0.0.0"
        imagePullPolicy: IfNotPresent
        rootLogLevel: INFO
        appLogLevel: INFO
    configEnv:
        name: "pdcrsm-configmap-env"
        rsmListenerPort: 8080
        baseURL: xxxxx.xxx.xxxxx.xxx
        useT3s: true
        securityEnabled: true
        securityType: oam
        resources:
            requests:
               cpu: "50m"
               memory: "256Mi"
            limits:
               cpu: "1000m"
               memory: "2Gi"
        oam:
            domainName: PDCRSMDomain
            audience: PDCRSMResourceServer
            endpointURL: http://oam_host:oam_port/oauth2/rest/token
            introspectendpointuri: http://oam_host:oam_port/oauth2/rest/token/info
            scopeaudience: http://oam_host:oam_port/
            authorizationendpointuri: http://oam_host:oam_port/oauth2/authorize
            proxyhost: http://proxyhost:proxyport/
            frontenduri: http://oam_host:oam_port
    secretValue:
        name: "pdcrsm-secret-env"
    service:
        name: "pdcrsm"
        type: "ClusterIP"
```

Configuring OAuth Authentication in PDC REST Services Manager

PDC REST Services Manager uses the OAuth 2.0 protocol to authenticate an enterprise product catalog's identity and to authorize the enterprise product catalog to access the PDC REST Services Manager API. It does this by validating an OAuth access token that is passed in the header of every HTTP/HTTPS request to the PDC REST Services Manager API.

To configure OAuth authentication in PDC REST Services Manager:

1. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart**:
   - If you are using Oracle Identity Cloud Service (IDCS) for OAuth:
     - **ocpdcrsm.configEnv.isInboundOauthEnabled**: Set this to **true** to enable OAuth authentication.
     - **ocpdcrsm.configEnv.inboundOauthUri**: Set this to the base URL of your Oracle Identity Cloud Service (IDCS) instance in this format:

       ```
       https://idcs-TenantID.identity.oraclecloud.com
       ```
     - **ocpdcrsm.configEnv.inboundOauthClientId**: Set this to the client ID of your confidential application.
     - **ocpdcrsm.secretValue.inboundOauthClientSecret**: Set this to the Base64-encrypted client secret obtained from your IDCS application.
     - **ocpdcrsm.configEnv.inboundOauthFrontendUri**: Set this to the base URL of your confidential application when run, such as http://myapp.example.com:8080.
     - **ocpdcrsm.configEnv.inboundOauthAudience**: Set this to the primary audience as provisioned for the PDC REST Services Manager application in IDCS.
     - **ocpdcrsm.configEnv.inboundOauthProxyHost**: Set this to the host name of your proxy server, if required.
     - **ocpdcrsm.configEnv.inboundOauthPubEventScope**: Set this to the name of the scope for accessing the TMF620 Publish Event endpoint for inbound OAuth authentication, such as **pubevent**.
     - **ocpdcrsm.configEnv.inboundOauthMetricsScope**: Set this to the name of the scope for accessing the metrics endpoint for inbound OAuth authentication, such as **metrics**.
   - If you are using Oracle Access Management for OAuth:
     - **ocpdcrsm.configEnv.oam.domainName**: Set this to the name of the OAuth identity domain created in Oracle Access Management for PDC REST Services Manager.
     - **ocpdcrsm.configEnv.oam.audience**: Set this to the name of the OAuth resource server created in Oracle Access Management for PDC REST Services Manager.
     - **ocpdcrsm.configEnv.oam.endpointURL**: Set this to the URL for requesting an OAuth token from Oracle Access Management.
     - **ocpdcrsm.configEnv.oam.introspectendpointuri**: Set this to the URL for validating an OAuth token from Oracle Access Management.
     - **ocpdcrsm.configEnv.oam.scopeaudience**: Set this to the primary audience for PDC REST Services Manager in the Oracle Access Management resource, used for error handling. This is the same as **ocpdcrsm.configEnv.oam.frontenduri**, ending with **/**.

- **ocpdcrsm.configEnv.oam.authorizationendpointuri**: The URL for authorizing role-based access. PDC REST Services Manager does not support role-based access, so this will not be used.
- **ocpdcrsm.configEnv.oam.proxyhost**: Set this to the URL for your Oracle Access Management proxy server, if needed.
- **ocpdcrsm.configEnv.oam.frontenduri**: Set this to the URL for of the OAuth client created in Oracle Access Management for PDC REST Services Manager.

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

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

3. Restart the PDC REST Services Manager pods. If downtime is not a concern, both pods can be deleted and re-created by running the following command. Otherwise, delete one pod at a time, waiting for its replacement pod to become "Running" before deleting the next one.

```
kubectl -n BrmNameSpace delete pods --selector=app.kubernetes.io/name=pdcrsm
```

Configuring Requests to the Enterprise Product Catalog

PDC REST Services Manager sends requests to the enterprise product catalog when calling the enterprise product catalog's REST API and when publishing acknowledgment notifications.

To configure PDC REST Services Manager to send requests to the enterprise product catalog:

1. Open the **override-values.yaml** file for **oc-cn-helm-chart**.
2. Edit the keys in the file based on the type of authentication required by your enterprise product catalog:
   - For OAuth 2.0 authentication, edit the keys in Table 8-3.

Table 8-3 OAuth 2.0 Keys

Key	Path in Values.yaml file	Description
tokenEndpoint	**ocpdcrsm.configEnv.httpClients.security.oauth2**	The endpoint used to retrieve a token from.
clientId	**ocpdcrsm.configEnv.httpClients.security.oauth2**	The client ID used to authenticate the request from PDC REST Services Manager.
scope	**ocpdcrsm.configEnv.httpClients.security.oauth2**	The scopes required by the enterprise product catalog.
grantType	**ocpdcrsm.configEnv.httpClients.security.oauth2**	The grant type to be used for the OAuth flow: **client_credentials** or **password**.
clientSecret	**ocpdcrsm.secretValue.httpClients.security.oauth2**	The encrypted client secret used to authenticate the request from PDC REST Services Manager.
password	**ocpdcrsm.secretValue.httpClients.security.oauth2**	The encrypted password required for accessing the enterprise product catalog.

   - For basic authentication, edit the keys in Table 8-4.

Table 8-4 basicAuth Keys

Key	Path in Values.yaml file	Description
username	**ocpdcrsm.configEnv.httpClients.security.basicAuth**	The user name required for accessing the enterprise product catalog.
password	**ocpdcrsm.secretValue.httpClients.security.basicAuth**	The password required for accessing the enterprise product catalog.

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

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

4. Restart the PDC REST Services Manager pods. If downtime is not a concern, both pods can be deleted and re-created by running the following command. Otherwise, delete one pod at a time, waiting for its replacement pod to have a "Running" status before deleting the next one.

   ```
   kubectl --namespace BrmNameSpace delete pods --selector=app.kubernetes.io/name=pdcrsm
   ```

The following shows an example configuration for OAuth 2.0 authentication.

Note:

All **urlRegex** values in the file must be properly escaped with \\. The characters that must be escaped are: **\.[]{}()<>*+-=!? ^$|**.

```
configEnv:
    httpClients:
        - urlRegex: "http://hostname:port/mobile/custom/catalogManagement/.*"
          security:
              oauth2:
                  tokenEndpoint: "https://idcs_hostname/oauth2/v1/token"
                  clientId: "fcb3443f6c504ed789ba38a78341b88a"
                  scope: "https://hostnameurn:opc:resource:consumer::all"
                  grantType: "password"
secretValue:
    httpClients:
        - urlRegex: "http://hostname:port/mobile/custom/catalogManagement/.*"
          security:
              oauth2:
                  clientSecret: client_secret
                  password: password
```

The following shows an example configuration for Basic authentication:

Note:

All **urlRegex** values in the file must be properly escaped with \\. The characters that must be escaped are: **\.[]{}()<>*+-=!? ^$|**.

```
configEnv:
    httpClients:
```

```
        - urlRegex: "http://hostname:port/mobile/custom/PublishingAPI.*"
          security:
              basicAuth:
                 username: eccUser
secretValue:
    httpClients:
        - urlRegex: "http://hostname:port/mobile/custom/PublishingAPI.*"
          security:
              basicAuth:
                  password: password
```

Enabling TLS in PDC REST Services Manager

You can enable TLS encryption in PDC REST Services Manager to secure the communications it receives from your enterprise product catalog.

To enable TLS in PDC REST Services Manager:

1. Generate a self-signed SSL certificate:
   a. Create a directory for storing your SSL certificate that is accessible by the BRM Helm chart, such as **oc-cn-helm-chart/rsm_cert**.
   b. Generate an SSL certificate. For example, this creates a certificate file named **cert.pem**:

      ```
      openssl req -x509 -newkey rsa:4096 -keyout openSSLKey.pem -out cert.pem -days
      365 -nodes
      ```

   c. Generate a PKCS12 KeyStore file. For example, this creates a KeyStore file named **keystore.p12**:

      ```
      openssl pkcs12 -export -out keyStore.p12 -inkey openSSLKey.pem -in cert.pem
      ```

2. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart**:
   - **ocpdcrsm.configEnv.isTlsEnabled**: Set this to **true** to enable TLS encryption for PDC REST Services Manager.
   - **ocpdcrsm.configEnv.tlsVersions**: Set this to the list of supported TLS versions, such as **TLSv1.2,TLSv1.3**.
   - **ocpdcrsm.configEnv.tlsCertificateFile**: Set this to the path to the TLS certificate bundle in the Helm chart. The certificate must be in PKCS12 format.
   - **ocpdcrsm.secretValue.tlsCertificatePassphrase**: Set this to the Base64-encrypted passphrase for the TLS certificate.
3. Run the **helm upgrade** command to update the Helm release:

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

4. To apply the changes, re-create any previously existing PDC REST Services Manager pods:

   ```
   kubectl --namespace BrmNameSpace delete pods --selector=app.kubernetes.io/name=pdcrsm
   ```

After you enable TLS, connect to PDC REST Services Manager services using HTTPS only.

Enabling T3S in PDC REST Services Manager

Enable the T3S protocol in PDC REST Services Manager to secure its communications to PDC.

To enable T3S in PDC REST Services Manager:

1. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart**:
   - **ocpdcrsm.configEnv.useT3s**: Set this to **true**.
2. Run the **helm upgrade** command to update the Helm release:

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

3. To apply the changes, re-create any previously existing PDC REST Services Manager pods:

   ```
   kubectl --namespace BrmNameSpace delete pods --selector=app.kubernetes.io/name=pdcrsm
   ```

Configuring Mapping of TMF620 priceType to BRM Events

If you are using PDC REST Services Manager, you must configure the mappings of BRM event names to the values your enterprise product catalog sends in the **priceType** property of the **ProductOfferingPrice** element of the TMF620 payload.

The mappings are configured in **configmap_pdcrsm_appeventCfg.yaml**. You can add mappings as needed for your deployment or use the default mappings provided at installation.

To add or edit mappings:

1. Open the **configmap_pdcrsm_appeventCfg.yaml** file.
2. Edit the existing mappings, or use them as templates to add new ones. Use the following format:

   ```
   pricetype : "eventname"
   ```

   where:
   - *pricetype* is the value sent in the **priceType** property of the **ProductOfferingPrice** element of the TMF620 payload.
   - *eventname* is the name of the BRM event the price type should be mapped to.

   For example, the default mappings for one-time fees and usage events are:

   ```
   ONE_TIME : "EventBillingProductFeePurchase"
   ONE_TIME_PRICE_PLAN : "EventBillingProductFeePurchase"
   USAGE : "EventSession"
   USAGE_PRICE_PLAN : "EventSession"
   ```

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

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

4. Restart the PDC REST Services Manager pods. If downtime is not a concern, both pods can be deleted and re-created by running the following command. Otherwise, delete one pod at a time, waiting for its replacement pod to become "Running" before deleting the next one.

   ```
   kubectl --namespace BrmNameSpace delete pods --selector=app.kubernetes.io/name=pdcrsm
   ```

9
Configuring the BRM Client Services

Learn how to configure Billing Care, Billing Care REST API, and Business Operations Center to run in your Oracle Communications Billing and Revenue Management (BRM) cloud native environment.

Topics in this document:

About Configuring Your BRM Client Services

Business Operations Center, Billing Care, and Billing Care REST API share a similar image stack.

Figure 9-1 shows the process for deploying Billing Care using WebLogic Operator. The same process is used for the Billing Care REST API. The only difference is the name of the deployer: **bcws-domain-deployer**.

Figure 9-1   Billing Care Deployment Flow

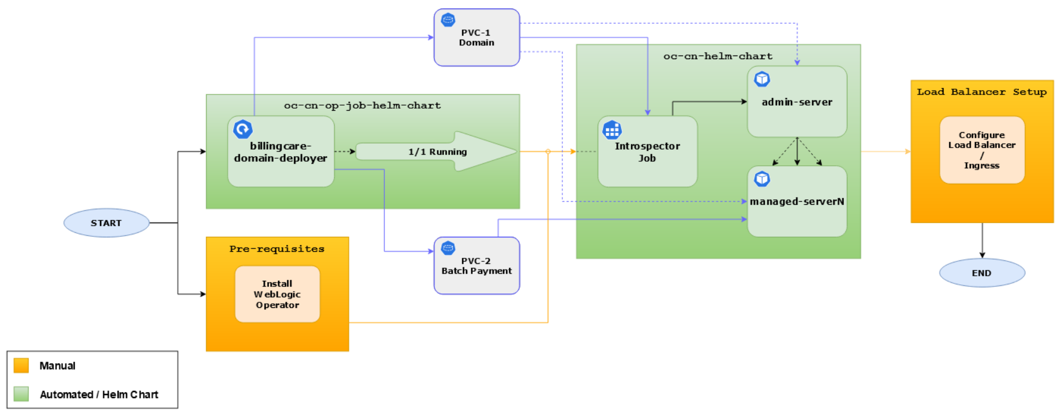
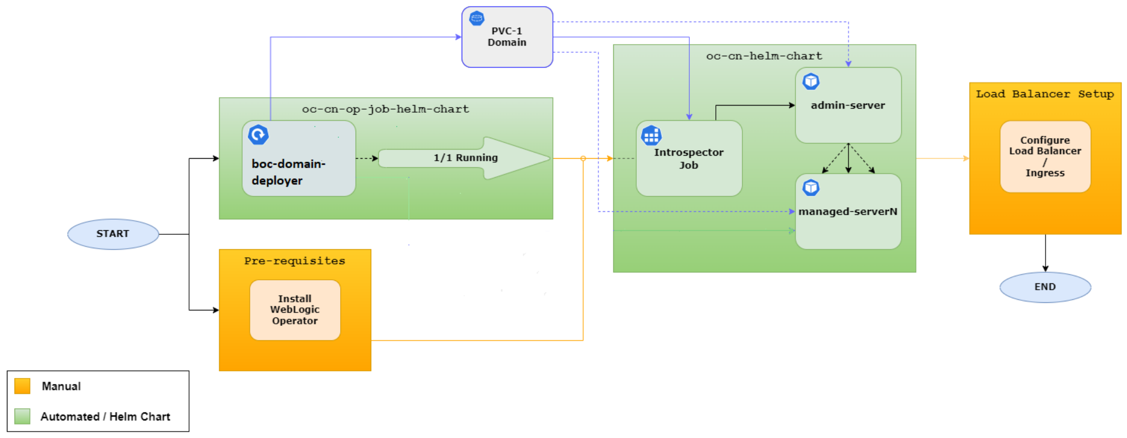


Figure 9-2 shows the process for deploying Business Operations Center using WebLogic Operator. It is similar to the Billing Care process.

Figure 9-2 Business Operations Center Deployment Flow



Note:

It is important to wait until the *component*-**domain-deployer** process is in the **1/1 Running** status before running **oc-cn-helm-chart**.

You deploy these services by using the following Helm charts:

- **oc-cn-op-job-helm-chart**: This chart creates and configures the WebLogic domain, deploys the application, deploys and links the SDK (for Billing Care and Billing Care REST API), and loads the authorization policies.
- **oc-cn-helm-chart**: This chart starts the rolling restart of the WebLogic servers and the application update.
- **WebLogic Operator chart**: This chart manages the application domain, controlling the service availability when managed server pods are scaled up or down.

Configuring Business Operations Center

Business Operations Center is a Web-based client application that you use to run business operations such as billing, invoicing, and payment collections. For more information, see "Using Business Operations Center" in *BRM System Administrator's Guide*.

To configure Business Operations Center to run in your BRM cloud native environment:

1. Override the Business Operations Center-specific keys in the **values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding Business Operations Center Keys for oc-cn-op-job-helm-chart".
2. Override the Business Operations Center-specific keys in the **values.yaml** file for **oc-cn-helm-chart**. See "Adding Business Operations Center Keys for oc-cn-helm-chart".
3. Set up volume mounts. See "About Business Operations Center Volume Mounts".
4. Create a WebLogic domain and install the Business Operations Center application. See "Creating a WebLogic Domain and Installing the Business Operations Center Application".
5. Set up SAML for SSO in Business Operations Center. See "Setting Up SSO for Business Operations Center".

6. Set up local users and groups for Business Operations Center. See "Setting Up Local Users and Groups for Business Operations Center".
7. Start and stop your WebLogic servers. See "Starting and Stopping WebLogic Servers".

Note:

To set up Business Operations Center, ensure that you successfully complete the installation of **oc-cn-op-job-helm-chart** before you install or upgrade **oc-cn-helm-chart**.

Adding Business Operations Center Keys for oc-cn-op-job-helm-chart

Table 9-1 lists the keys that directly impact Business Operations Center. Add these keys to your **override-values.yaml** file for **oc-cn-op-job-helm-chart** with the same path hierarchy.

For a complete set of keys to personalize Business Operations Center deployment, see the keys with the path **ocboc.boc** in the **oc-cn-op-job-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocboc.boc.secretVal** hold sensitive data. Handle them carefully with controlled access to the file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-1 Keys for oc-cn-op-job-helm-chart

Key	Path in Values.yaml file	Description
isEnabled	**ocboc.boc**	Whether to deploy, configure, and start Business Operation Center services. • **false**: Kubernetes resources meant for the Business Operation Center application will not be created. • **true**: Creates the necessary Kubernetes resources for using Business Operation Center. This is the default.
imageTag	**ocboc.boc.deployment.app**	The tag associated with the image. This is generally the release number prefixed with a colon (**:**). For example: **:15.1.0.0.0**.
rcuDBARole	**ocboc.boc.configEnv**	The role of the database administrator user.
rcuArgs	**ocboc.boc.configEnv**	The additional arguments for creating the RCU.

Table 9-1 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in Values.yaml file	Description
dbURL	**ocboc.boc.configEnv**	Used to create the WebLogic data source for connecting to the Business Operations Center schema. This is also the connection string for the database where schemas needed by Oracle Fusion Middleware products are created, especially OPSS. Use one of these formats: • *DatabaseHost***:***DatabasePortlServiceName* • *DatabaseHost***:***DatabasePort***:***ServiceID*
dbSSLMode	**ocboc.boc.configEnv**	The type of connection required to connect to the database: • **Yes-Two Way**: Two-way SSL authentication is required. • **Yes-One Way**: One-way SSL authentication is required. This is the default. • **No**: SSL authentication is not required.
dbWalletType	**ocboc.boc.configEnv**	The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**.
extDBSSLWalletSecret **extKeystoreSecret**	**ocboc.boc.configEnv**	The names of the pre-created Kubernetes Secrets for the Business Operations Center KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
dbWalletPassword	**ocboc.boc.secretVal**	The password for accessing the certificates from the TrustStore and KeyStore.
ldapHost	**ocboc.boc.configEnv**	The host name or IP address of the LDAP Server (for example, OUD) where users and groups are configured for access to Business Operations Center.
ldapPort	**ocboc.boc.configEnv**	The port number on which the LDAP server is listening.
ldapGroupBase	**ocboc.boc.configEnv**	The LDAP base DN that contains groups.
ldapUserBase	**ocboc.boc.configEnv**	The LDAP base DN that contains users.
bocSchemaUserName	**ocboc.boc.configEnv**	The Business Operations Center database schema user name.
bocSchemaBocTablespace	**ocboc.boc.configEnv**	The default tablespace for the Business Operations Center database administrator.
bocSchemaTempTablespace	**ocboc.boc.configEnv**	The temp tablespace for the Business Operations Center database administrator.
billingCareUrl	**ocboc.boc.configEnv**	The URL of the Billing Care instance that is used with your BRM Server. Leave this blank if Billing Care isn't installed in your environment.

Table 9-1 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in Values.yaml file	Description
isSSOEnabled	**ocboc.boc.configEnv**	Whether to enable single sign-on (SSO) for Business Operations Center cloud native services using SAML 2.0: • **true**: SSO is enabled for Business Operations Center cloud native services. • **false**: SSO is disabled. This is the default.
keystoreAlias	**ocboc.boc.configEnv**	The private key alias of the KeyStore.
keystoreType	**ocboc.boc.configEnv**	The file type of the SSL Identity and Trust store, which is either **PKCS12** or **JKS**. The default is **PKCS12**.
keystoreIdentityFileName	**ocboc.boc.configEnv**	The file name of the Identity KeyStore.
keystoreTrustFileName	**ocboc.boc.configEnv**	The file name of the Trust KeyStore.
tlsVersions	**ocboc.boc.configEnv**	The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**.
samlAsserterName	**ocboc.boc.configEnv**	The name of the SAML Asserter. The default is **samlBOCAsserter**.
ssoPublishedSiteURL	**ocboc.boc.configEnv**	The base URL that is used to construct endpoint URLs. This is typically the Load Balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
ssoDefaultURL	**ocboc.boc.configEnv**	The URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
reloadVersion	**ocboc.boc.configEnv**	Update this value with any value different from the current value to force a restart of the deployer.
adminPassword	**ocboc.boc.secretVal**	The Base64-encoded password for the WebLogic domain's administrative user. This is used for accessing the WebLogic Server Administration Console for administrative operations.
ldapPassword	**ocboc.boc.secretVal**	The Base64-encoded password of the LDAP Server admin user.
rcuSysDBAPassword	**ocboc.boc.secretVal**	The Base64-encoded database administrator's password.
bocSchemaPassword	**ocboc.boc.secretVal**	The Base64-encoded Business Operations Center database schema password.
rcuSchemaPassword	**ocboc.boc.secretVal**	The Base64-encoded password for schemas of Oracle Fusion Middleware products that will be created by RCU, which is used by OPSS.
keystoreIdentityPassword	**ocboc.boc.secretVal**	The StorePass for the Identity KeyStore.
keystoreKeyPassword	**ocboc.boc.secretVal**	The KeyPass for the Identity KeyStore.
keystoreTrustPassword	**ocboc.boc.secretVal**	The StorePass for the Trust KeyStore.

Table 9-1 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in Values.yaml file	Description
domainUID	**ocboc.boc.wop**	The name of the domain. The default is **boc-domain**.
adminChannelPort	**ocboc.boc.wop**	The NodePort where the admin-server's HTTP service is accessible.
serverStartPolicy	**ocboc.boc.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count.
domain.*	**ocbc.boc.volume**	Details about the PVC for the domain file system: • **storage**: The storage size of the volume. • **createOption**: By default, the boc pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocboc.boc**	The rules for scheduling WebLogic Server pods on particular nodes using simple selectors using Node Selector rules.
affinity	**ocboc.boc**	The rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors using affinity rules.

Adding Business Operations Center Keys for oc-cn-helm-chart

Table 9-2 lists the keys that directly impact Business Operations Center. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart** with the same path hierarchy.

For a complete set of keys to personalize Business Operations Center deployment, see the keys with the path **ocboc.boc** in the **oc-cn-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocboc.boc.secretVal** hold sensitive data. Handle them carefully with controlled access to the file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-2 Keys for oc-cn-helm-chart

Key	Path in Values.yaml file	Description
isEnabled	**ocboc.boc**	Whether to deploy, configure, and start Business Operation Center services. • **false**: Kubernetes resources meant for the Business Operation Center application will not be created. • **true**: Creates the necessary Kubernetes resources for using Business Operation Center. This is the default.
imageTag	**ocboc.boc.deployment .app**	The tag associated with the image. This is generally the release number prefixed with a colon (**:**). For example, **:15.1.0.0.0**.
login	**ocboc.boc.infranet.use r**	The user name of the service with permission to access BRM, such as **boc_client.0.0.0.1**.
serviceType	**ocboc.boc.infranet.use r**	The POID type of the service that has permission to access BRM.
serviceID	**ocboc.boc.infranet.use r**	The POID ID of the service that has permission to access BRM.
minSize	**ocboc.boc.infranet.con nectionpool**	The minimum number of connections allowed in the pool. The default is **25**.
maxSize	**ocboc.boc.infranet.con nectionpool**	The maximum number of connections allowed in the pool. The default is **50**.
loglevel	**ocboc.boc.infranet**	The log level for the **Infranet.properties** file.
addOnProperties	**ocboc.boc.infranet**	This field is empty by default. You can use this key to specify custom **Infranet.properties** values.
domainUID	**ocboc.boc.wop**	The name of the domain. The default is **boc-domain**.
adminChannelPort	**ocboc.boc.wop**	The NodePort where the admin-server's http service will be accessible. The default is empty. **Note:** Set this key only if you want the **boc-domain-admin-server-ext** service to deploy as NodePort.
serverStartPolicy	**ocboc.boc.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count.
isEnabled	**ocboc.boc.monitoring**	Whether to enable monitoring of Business Operations Center. See "Monitoring and Autoscaling Business Operations Center Cloud Native" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocboc.boc**	The rules for scheduling WebLogic Server pods on particular nodes using simple selectors using Node Selector rules.

Table 9-2 (Cont.) Keys for oc-cn-helm-chart

Key	Path in Values.yaml file	Description
affinity	**ocboc.boc**	The rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors using affinity rules.
extDBSSLWalletSecret **extKeystoreSecret**	**ocboc.boc.configEnv**	The names of the pre-created Kubernetes Secrets for the Business Operations Center KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.

Updating Infranet.properties for Business Operations Center

The **Infranet.properties** file entries are located in the **values.yaml** file. This makes it easier to update them.

Following is a sample configuration block (located in the **ocboc.boc** path in **oc-cn-helm-chart**) for the **Infranet.properties** entries:

```
infranet:
    user:
        login: 'boc_client.0.0.0.1'
        serviceType: '/service/admin_client'
        serviceId: 2
    connectionpool:
        minSize: 25
        maxSize: 50
    logLevel: 3
    addOnProperties: ""
```

If you have custom properties, they should be defined here using the **addOnProperties** key. For example:

```
addOnProperties: |-
    infranet.connectionpool.timeout=90000
    infranet.pcp.debug.flags=0x3FFF
    infranet.pcp.debug.enabled=true
```

To update these properties, update the values in **oc-cn-helm-chart** and change the value of **ocboc.boc.wop.restartVersion** in **oc-cn-helm-chart** to any new value. This will force a pod restart and the new values will be used.

Adding Custom Configuration to Deployment Workflow for Business Operations Center

You can provide additional configuration to be applied at particular checkpoints in the Business Operations Center deployment workflow. These checkpoints are:

- **ext_deployer_pre_exit**: Called after the standard configuration in **deployer.sh** in **oc-cn-op-job-helm-chart**
- **ext_init_app_pre_exit**: Called after the standard configuration in the init-app **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**

- **ext_init_config_pre_exit**: Called after the standard configuration in the init-config **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**
- **ext_init_upgrade_pre_exit**: Called after the standard configuration in the **upgrade** container

Create a ConfigMap with your configuration scripts, including a shell script named **run_hooks.sh** that calls your other scripts. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ext-scripts
data:
  run_hooks.sh: |+
    #!/bin/bash
    echo "executing extension for: $@"
    CURRENT_CHECKPOINT=$1
    if [ "$CURRENT_CHECKPOINT" == "ext_deployer_pre_exit" ] ; then
      sh my_deployer_extension.sh
    fi
  my_deployer_extension.sh: |+
    #!/bin/bash
    echo "executing my_deployer_extension"
...
```

Specify the name of your ConfigMap in the **ocboc.boc.extensions.scriptsConfigName** key in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

About Business Operations Center Volume Mounts

The Business Operations Center container requires Kubernetes volume mounts for sharing the domain and application file system between the WebLogic Cluster servers. Business Operations Center requires a volume for the domain. By default, this is created dynamically, using the provisioner defined in BRM, in the **storage-class** key in **oc-cn-op-job-helm-chart**.

To change the volume type or provider, modify the **ocboc.boc.volume.domain.createOption** key in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

Creating a WebLogic Domain and Installing the Business Operations Center Application

The WebLogic domain is created by a Kubernetes Deployment when **oc-cn-op-job-helm-chart** is installed. The same job also installs the Business Operations Center application and deploys the application WAR file onto the WebLogic Cluster.

The **oc-cn-op-job-helm-chart** chart also:

- Creates a Kubernetes ConfigMap and Secrets, which are used throughout the life-cycle of the WebLogic domain.
- Initializes the **PersistentVolumeClaim** for the domain and application file system as well as third-party libraries.

Note:

The **override-values.yaml** file that you use for this chart must include BRM override values.

After you install **oc-cn-op-job-helm-chart**, wait until the Kubernetes deployment has reached the **1/1 Running** status. Then, you can install or upgrade **oc-cn-helm-chart** for Business Operations Center services.

After the deployment is running, don't delete the chart. Its resources will be used for starting and stopping the servers through **oc-cn-helm-chart**.

Setting Up SSO for Business Operations Center

SSO allows users to log in to applications using a single user name and password combination. You set up SSO for Business Operations Center cloud native services by using SAML 2.0.

To set up SSO for Business Operations Center:

1. Export the SAML 2.0 metadata XML file from your identity and access management (IAM) system.

   For example, if you are using Oracle Access Management, you can export the file by following the instructions in "Exporting Metadata" in *Oracle Fusion Middleware Administering Oracle Access Management*.

2. Add the metadata XML file to your BRM cloud native deployment by doing one of the following:

   - Rename the metadata XML file to **metadata.xml**, and then move **metadata.xml** to the **oc-cn-op-job-helm-chart/boc/idp** directory. In this case, you must leave the **ocboc.boc.configEnv.extMetadataCM** key for **oc-cn-op-job-helm-chart** empty.
   - Pre-create the IDP metadata ConfigMap for Business Operations Center and set the **ocboc.boc.configEnv.extMetadataCM** key in your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

3. Configure the KeyStores needed by SAML by doing one of the following:

   - Generate the Identity and Trust KeyStores and then move your files, such as **identity.p12** and **trust.p12**, under the **oc-cn-op-job-helm-chart/boc/keystore** directory. In this case, you must leave the **ocboc.boc.configEnv.extKeystoreSecret** key for **oc-cn-op-job-helm-chart** empty.
   - Pre-create the Kubernetes Secret for the Identity and Trust KeyStore files and set the **ocboc.boc.configEnv.extKeystoreSecret** key in your **override-values.yaml** file for both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

4. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, set the following keys:

   - **ocboc.boc.configEnv.isSSOEnabled**: Set this to **true**.
   - **ocboc.boc.configEnv.keystoreAlias**: Set this to the private key alias of the KeyStore.

- **ocboc.boc.configEnv.keystoreType**: Set this to the file type of the SSL Identity and Trust store, which is either **PKCS12** or **JKS**. The default is **PKCS12**.
- **ocboc.boc.configEnv.keystoreIdentityFileName**: Set this to the name of the Identity KeyStore file.
- **ocboc.boc.configEnv.keystoreTrustFileName**: Set this to the name of the Trust KeyStore file.
- **ocboc.boc.configEnv.samlAsserterName**: Set this to the name of the SAML Asserter. The default is **samlBOCAsserter**.
- **ocboc.boc.configEnv.ssoPublishedSiteURL**: Set this to the base URL that is used to construct endpoint URLs. This is typically the load balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
- **ocboc.boc.configEnv.ssoDefaultURL**: Set this to the URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
- **ocboc.boc.secretVal.keystoreIdentityPassword**: Set this to the StorePass for the Identity KeyStore.
- **ocboc.boc.secretVal.keystoreKeyPassword**: Set this to the KeyPass for the Identity KeyStore.
- **ocboc.boc.secretVal.keystoreTrustPassword**: Set this to the StorePass for the Trust KeyStore.

5. Configure your load balancer's rules to send responses to the Business Operations Center WebLogic domain with **/saml2** appended to the URL path.

   Note:

   Add this rule to your existing load balancer rules for routing responses to Business Operations Center (**/opsdashboard**), the host name, and so on.

   See "Installing an Ingress Controller".

6. Deploy your Business Operations Center cloud native services by following the instructions in "Deploying BRM Cloud Native Services".

7. After Business Operations Center is deployed, retrieve the **sp-metadata-admin-server.xml** file from the **/shared/domains/***domainUID* directory in your container, where *domainUID* is the name of your Business Operations Center domain specified in the **ocboc.boc.wop.domainUID** key.

   The XML file configures the Web SSO Provider Partner. It contains the partner's KeyStore certificates, SAML assertion details, and the URLs where the SAML Identity Provider redirects to provide access to Business Operations Center.

8. Create a profile for your identity provider partner by loading the **sp-metadata-admin-server.xml** file into your IAM system.

   For example, if you are using Oracle Access Management, you can load the file by following the instructions in "Creating Remote Identity Provider Partners" in *Oracle Fusion Middleware Administering Oracle Access Management*.

Setting Up Local Users and Groups for Business Operations Center

You have the option to customize the values for **oc-cn-op-job-helm-chart** to create users and groups locally in Oracle WebLogic Server. This would be especially useful for test environments where you might not have Identity Providers or LDAPs available. The groups for the admin user for WebLogic Server cannot be modified using this procedure.

Any passwords must be encoded using Base64. You can leave the password blank, but then the user will not be able to log in to the application directly.

To set up local users and groups for Billing Care, define the keys under **ocboc.boc.wlsUserGroups** in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**. For example:

```
ocboc:
    boc:
        wlsUserGroups:
            groups:
            -   name: "GroupA"
                description: "GroupA Description"
            -   name: "GroupB"
                description: "GroupB Description"
            users:
            -   name: csr1
                description: "csr1 description"
                password: "Base64_password"
                groups:
                -   "GroupA"
                -   "GroupB"
            -   name: csr2
                description: "csr2 description"
                password: "Base64_password"
                groups:
                -   "GroupB"
```

Starting and Stopping WebLogic Servers

When you install **oc-cn-op-job-helm-chart**, the default configuration sets up a WebLogic Cluster with five Managed Servers. When you install or upgrade **oc-cn-helm-chart** for the Business Operations Center service, two of the managed servers and one Admin Server are started.

By modifying the **override-values.yaml** file for **oc-cn-helm-chart**, you can control:

- The total number of Managed Servers and the initial server start up by using the **totalManagedServers** and **initialServerCount** keys.
- Whether the servers are started or stopped by using the **serverStartPolicy** key. To start the Admin Servers and the Managed Servers in a Cluster, set the key to **IF_NEEDED**. To stop all servers, set the key to **NEVER**.

Note:

The keys in the **override-values.yaml** file should be the same as the ones used in **oc-cn-op-job-helm-chart** for keys that are common in both charts.

Before installing or upgrading **oc-cn-helm-chart** for Business Operations Center, ensure that the brm_apps values are configured correctly. If there is a change in any brm_apps values, use **serverStartPolicy** to restart and have the changes take effect.

After you modify the **override-values.yaml** file, update the Helm release for the changes to take effect:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

Configuring Billing Care

Billing Care is a Web-based client application that CSRs use to manage billing, payments, and accounts receivable for your customers. For more information about using Billing Care, see *Billing Care Online Help*.

To configure Billing Care to run in your BRM cloud native environment:

1. Override the Billing Care-specific keys from the **values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding Billing Care Keys for oc-cn-op-job-helm-chart".
2. Override the Billing Care-specific keys from the **values.yaml** file for **oc-cn-helm-chart**. See "Adding Billing Care Keys for oc-cn-helm-chart".
3. Set up volume mounts for Billing Care. See "About Billing Care Volume Mounts".
4. Create a WebLogic domain and install Billing Care. See "Creating a WebLogic Domain and Installing the Billing Care Application".
5. Set up SAML for SSO in Billing Care. See "Setting Up SSO for Billing Care".
6. Set up local users and groups for Billing Care. See "Setting Up Local Users and Groups for Billing Care".
7. Start and stop your WebLogic servers. See "Starting and Stopping WebLogic Servers".

Note:

To set up Billing Care, ensure that you successfully complete the installation of **oc-cn-op-job-helm-chart** before you install or upgrade **oc-cn-helm-chart**.

Adding Billing Care Keys for oc-cn-op-job-helm-chart

Table 9-3 lists a few important keys that directly impact Billing Care. Add these keys to your **override-values.yaml** file for **oc-cn-op-job-helm-chart** with the same path hierarchy.

For the complete set of keys to personalize your Billing Care deployment, see the keys with the path **ocbc.bc** in the **oc-cn-op-job-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocbc.bc.secretVal** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-3 Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
isEnabled	**ocbc.bc**	Whether to deploy, configure, and start Billing Care services (**true**) or not (**false**). The default is **true**.
deployment.app.*	**ocbc.bc**	The details about the Billing Care application image. • **imageName**: The name of the Billing Care image, such as **oracle/billingcare**. • **imageTag**: The tag associated with the image. This is generally the release number prefixed with a colon (**:**). For example, **:15.1.0.0.0**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.
deployment.fmw.*	**ocbc.bc**	The details about the Fusion Middleware Infrastructure image. • **imageRepository**: The name of the repository from where the Fusion Middleware Infrastructure image is pulled. The default is **container-registry.oracle.com/**. • **imageName**: The name of the Fusion Middleware Infrastructure image, such as **middleware/fmw-infrastructure**. • **imageTag**: The tag associated with the image. For example: **:14.1.2.0-jdk21-ol9**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
deployment.sdk.*	**ocbc.bc**	The details about the Billing Care SDK image. • **imageName**: The name of the Billing Care SDK image. • **imageTag**: The tag associated with the image. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.
sdk.*	**ocbc.bc**	The details for deploying the Billing Care SDK. • **isEnabled**: Whether to deploy your SDK customizations for overriding application behavior (**true**) or not (**false**). The default is **false**. • **deployName**: The name of the Billing Care SDK to appear in the deployment list.
configEnv.*	**odbc.bc**	The details about the Managed Server. • **managedHttpPort**: The container's port for access to the Managed Server. The default is **8001**. • **httpPort**: The container's port for access to the WebLogic Domain over HTTP. The default is **7011**. • **serverStartMode**: The mode to use when starting the server: development mode (**dev**) or production mode (**prod**). The default is **prod**. • **adminUser**: The user who will be granted administrator rights to the WebLogic Domain. • **dbSSLMode**: The type of connection required to connect to the database: one-way SSL authentication (**ONE_WAY**) or SSL authentication is not required (**no**). • **dbWalletType**: The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**. • **extDBSSLWalletSecret**: The names of the pre-created Kubernetes Secrets for the Billing Care KeyStore certificate file. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
RCU Schema	**ocbc.bc.configEnv**	The details about the RCU schema. • **rcuJdbcURL**: The connection string for the database where schemas needed by Oracle Fusion Middleware products will be created, especially OPSS. • **rcuSysDBAUser**: The database administrator user name. • **rcuDBARole**: The role of the database administrator user. • **rcuPrefix**: The prefix for the OPSS schema. The default is **BC01**. • **rcuRecreate**: Whether to drop the existing OPSS schema (**true**) or not (**false**). The default is **true**. • **rcuTablespace**: The name of an existing tablespace in your database. If left empty, new tablespaces are created with names starting with **rcuPrefix**. • **rcuTempTablespace**: The name of an existing temporary tablespace in your database. If left empty, new tablespaces are created with names starting with **rcuPrefix**. • **isOPSS**: Whether to create an OPSS domain (**true**) or a non-OPSS domain (**false**). The default is **true**. • **extAccessPolicyCM**: The name of the ConfigMap containing the policy file.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
LDAP Server	**ocbc.bc.configEnv**	The details about the LDAP Server. • **isLDAPEnabled**: Whether to skip creation of the Oracle Unified Directory Authenticator (**true**) or not (**false**). The default is **true**. • **ldapAdmin**: The Distinguished Name to connect to the LDAP server. • **ldapHost**: The host name or IP address of the LDAP Server (for example, OUD) where users and groups will be configured for access to Billing Care. • **ldapPort**: The port number on which the LDAP server is listening. • **ldapGroupBase**: The LDAP base DN that contains groups. • **ldapUserBase**: The LDAP base DN that contains users. • **ldapProviderName**: The name of Authentication Provider. • **targetServer**: The server in the WebLogic domain where the application must be deployed.
KeyStore Certificates	**ocbc.bc.configEnv**	The details about the KeyStore certificates for Billing Care. • **keystoreAlias**: The private key alias of the KeyStore. • **extKeystoreSecret**: The name of the pre-created external Secret containing the Identity and Trust KeyStore certificate file. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **keystoreType**: The file type of the SSL Identity and TrustStore certificate, which is either **PKCS12** or **JKS**. The default is **PKCS12**. • **keystoreIdentityFileName**: The file name of the Identity KeyStore certificate file. • **keystoreTrustFileName**: The file name of the TrustStore certificate file.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
Secure Connections	**ocbc.bc.configEnv**	The details for secure connections. • **tlsVersions**: The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**. • **isSSOEnabled**: Whether to enable single sign-on (SSO) for Billing Care cloud native services through SAML 2.0 (**true**) or SSO is disabled (**false**). The default is **false**. • **extMetadataCM**: The name of the ConfigMap containing the IDP metadata file. • **samlAsserterName**: The name of the SAML Asserter. The default is **samlBCAsserter**. • **ssoPublishedSiteURL**: The base URL that is used to construct endpoint URLs. This is typically the Load Balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**. • **ssoDefaultURL**: The URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL. • **reloadVersion**: Set this to any value different from the current value to force a restart of the deployer. The default is **1**. • **reset**: Whether to wipe all previous states and do a fresh setup of the domain (**true**) or not (**false**). The default is **false**. When set to **true**, you must change the **introspectVersion** key in the **oc-cn-helm-chart** must after upgrading **oc-cn-op-job-helm-chart**.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
secretValue.*	**ocbc.bc**	The credentials for accessing the system. • **adminPassword**: The password of the WebLogic domain's administrative user, which is used for accessing the WebLogic Console for administrative operations. • **ldapPassword**: The password of the LDAP Server admin user. • **rcuSysDBAPassword**: The password for the rcuJdbcURL database administrator. • **rcuSchemaPassword**: The passwords for the schemas of Oracle Fusion Middleware products that will be created by RCU, which is used by OPSS. • **dbWalletPassword**: The password for accessing the certificates from the TrustStore and KeyStore. • **keystoreIdentityPassword**: The StorePass for the Identity KeyStore. • **keystoreKeyPassword**: The KeyPass for the Identity KeyStore. • **keystoreTrustPassword**: The StorePass for the Trust KeyStore.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
wop.*	**ocbc.bc**	The details about the WebLogic Domain. • **domainUID**: The name of the domain, which is used as a prefix to tag related objects. The default is **billingcare-domain**. • **domainRootDir**: The location within the container where the domain is created. The default is **/shared**. • **totalManagedServers**: The total number of managed servers forming the cluster. The default is **5**. • **initialServerCount**: The number of managed servers to initially start for the domain. The default is **2**. • **adminChannelPort**: The NodePort where the admin-server's HTTP service is accessible. The default is **30721**. • **serverStartPolicy**: The WebLogic servers that the Operator starts when it discovers the domain: – **NEVER**: Does not start any server in the domain. – **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). – **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count. This is the default.
domain.*	**ocbc.bc.volume**	Details about the PVC for the domain file system. • **storage**: The storage size of the volume. • **createOption**: By default, the billingcare pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 9-3 (Cont.) Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
batchPayment.*	**ocbc.bc.volume**	Details about the PVC for the batch payment files. • **storage**: The storage size of the volume. • **createOption**: By default, the billingcare pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
wlsUserGroups.*	**ocbc.bc**	The details for adding users and groups to the domain's default authenticator.
extensions.*	**ocbc.bc**	The name of the ConfigMap containing scripts to execute additional steps to configure the domain and application.
resources.*	**ocbc.bc**	The minimum and maximum CPU and memory resources that containers can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocbc.bc**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocbc.bc**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.
addOnPodSpec	**ocbc.bc.pod**	The add details to inject into the pod specification.

Adding Billing Care Keys for oc-cn-helm-chart

Table 9-4 lists a few important keys that directly impact Billing Care. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart** with the same path hierarchy.

For the complete set of keys to personalize your Billing Care deployment, see the keys with the path **ocbc.bc** in the **oc-cn-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocbc.bc.secretVal** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-4 Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
appLogLevel	**ocbc**	The logging level at which application logs must be captured in log files: **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**, and **ALL**.
isEnabled	**ocbc.bc**	Whether to deploy, configure, and start Billing Care services (**true**) or not (**false**). The default is **true**.
deployment.app.*	**ocbc.bc**	The details about the Billing Care application image. • **imageName**: The name of the Billing Care image, such as **oracle/billingcare**. • **imageTag**: The tag associated with the image. This is generally the release number prefixed with a colon (**:**). For example, **:15.1.0.0.0**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.
deployment.fmw.*	**ocbc.bc**	The details about the Fusion Middleware Infrastructure image. • **imageRepository**: The name of the repository from where the Fusion Middleware Infrastructure image is pulled. The default is **container-registry.oracle.com/**. • **imageName**: The name of the Fusion Middleware Infrastructure image, such as **middleware/fmw-infrastructure**. • **imageTag**: The tag associated with the image. For example: **:14.1.2.0-jdk21-ol9**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.
deployment.sdk.*	**ocbc.bc**	The details about the Billing Care SDK image. • **imageName**: The name of the Billing Care SDK image. • **imageTag**: The tag associated with the image. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**.

Table 9-4 (Cont.) Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
sdk.*	**ocbc.bc**	The details for deploying the Billing Care SDK. • **isEnabled**: Whether to deploy your SDK customizations for overriding application behavior (**true**) or not (**false**). The default is **false**. • **deployName**: The name of the Billing Care SDK to appear in the deployment list.
configEnv.*	**odbc.bc**	The details about the Managed Server. • **httpPort**: The container's port for access to the WebLogic Domain over HTTP. The default is **7011**. • **isOPSS**: Whether to create an OPSS domain (**true**) or a non-OPSS domain (**false**). The default is **true**. • **keystoreAlias**: The private key alias of the KeyStore. • **extKeystoreSecret**: The name of the pre-created external Secret containing the Identity and Trust KeyStore certificate file. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **keystoreType**: The file type of the SSL Identity and TrustStore certificate, which is either **PKCS12** or **JKS**. The default is **PKCS12**. • **dbSSLMode**: The type of connection required to connect to the database: one-way SSL authentication (**ONE_WAY**) or SSL authentication is not required (**no**). • **dbWalletType**: The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**. • **extDBSSLWalletSecret**: The names of the pre-created Kubernetes Secrets for the Billing Care KeyStore certificate file. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*. • **bipUrl**: The URL to the BI Publisher server. • **bipUserId**: The name of the user with access to the BI Publisher instance.

Table 9-4 (Cont.) Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
infranet.*	**ocbc.bc**	The details for connecting to BRM. • **user.***: Information about the user having permissions to access BRM. • **connectionpool.***: The details about the connection pool. The default minimum is **25** and the default maximum is **50**. • **loglevel**: The log level for the **infranet.properties** file. • **ssoLogoutURL**: The URL where the user is redirected after logging out from the application. • **addOnProperties**: This key is empty by default. You can use this key to specify custom **Infranet.properties** values.
secretVal.*	**ocbc.bc**	The credentials for accessing the system. • **walletPassword**: The password of the wallet storing sensitive data for the BRM connection. • **bipPassword**: The password of the BI Publisher instance. • **keystoreTrustPassword**: The StorePass for the Trust KeyStore.

Table 9-4 (Cont.) Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
wop.*	**ocbc.bc**	The details about the WebLogic Domain. • **domainUID**: The name of the domain, which is used as a prefix to tag related objects. The default is **billingcare-domain**. • **domainRootDir**: The location within the container where the domain is created. The default is **/shared**. • **totalManagedServers**: The total number of managed servers forming the cluster. The default is **5**. • **initialServerCount**: The number of managed servers to initially start for the domain. The default is **2**. • **adminChannelPort**: The NodePort where the admin-server's HTTP service is accessible. **Note:** Set this key only if you want the **billingcare-domain-admin-server-ext** service to deploy as NodePort. • **serverStartPolicy**: The WebLogic servers that the Operator starts when it discovers the domain: – **NEVER**: Does not start any server in the domain. – **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). – **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count. This is the default. • **restartVersion**: Whether to force a rolling restart of all server pods. Change to any value other than current to trigger the action. • **introspectVersion**: Whether to force a domain introspection on change in domain configuration. Change to any value other than current to trigger the action.

Table 9-4 (Cont.) Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
monitoring.*	**ocbc.bc.volume**	Details about monitoring Billing Care. • **isEnabled**: Whether to enable monitoring of Billing Care (**true**) or not (**false**). See "Monitoring and Autoscaling Billing Care Cloud Native" in *BRM Cloud Native System Administrator's Guide*. • **imageRepository**: By default, the billingcare pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. • **imageName**: The name of the WebLogic Monitoring Exporter image. The default is **oracle/weblogic-monitoring-exporter**. • **imageTag**: The tag associated with the image. The default is **:2.2.2**. • **imagePullPolicy**: When to pull images: only when one is not present locally (**IfNotPresent**) or always (**Always**). The default is **IfNotPresent**. • **scrapeInterval**: The duration at which Prometheus scrapes the target. The default is **2s**. • **operator.isEnabled**: Whether the system is using Prometheus Operator and ServiceMonitor to scrape metrics (**true**) or not (**false**). The default is **false**. • **resources.***: The minimum and maximum CPU and memory resources that containers can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
extensions.*	**ocbc.bc**	The name of the ConfigMap containing scripts to execute additional steps to configure the domain and application.
resources.*	**ocbc.bc**	The minimum and maximum CPU and memory resources that containers can use. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.

Table 9-4 (Cont.) Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
nodeSelector	**ocbc.bc**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocbc.bc**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.
serverPod	**ocbc.bc**	The add details to inject into the pod specification.

Updating Infranet.properties for Billing Care

The **Infranet.properties** file entries are located in the **values.yaml** file. This makes it easier to update them.

Following is a sample configuration block (located in the **ocbc.bc** path in **oc-cn-helm-chart**) for the **Infranet.properties** entries:

```
infranet:
    user:
        login: 'boc_client.0.0.0.1'
        serviceType: '/service/admin_client'
        serviceId: 2
    connectionpool:
        minSize: 25
        maxSize: 50
    logLevel: 3
    ssoLogoutURL:
    addOnProperties: ""
```

If you have custom field classes, they should be provided through the SDK **.war** file and defined here using the **addOnProperties** key. For example:

```
addOnProperties:|-
    infranet.custom.field.package=com.portal.custom
    infranet.custom.field.100011=PIN_FLD_ABC
```

To update these properties, update the values in **override-values.yaml** file for **oc-cn-helm-chart**. If this is an upgrade, also update the **ocbc.bc.wop.restartVersion** key in the same file. This will force a pod restart and the new values will be used.

Adding Custom Configuration to Deployment Workflow for Billing Care

You can provide additional configuration to be applied at particular checkpoints in the Billing Care deployment workflow. These checkpoints are:

- **ext_deployer_pre_exit**: Called after the standard configuration in **deployer.sh** in **oc-cn-op-job-helm-chart**
- **ext_init_app_pre_exit**: Called after the standard configuration in the init-app **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**
- **ext_init_config_pre_exit**: Called after the standard configuration in the init-config **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**

Create a ConfigMap with your configuration scripts, including a shell script named **run_hooks.sh** that calls your other scripts. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ext-scripts
data:
  run_hooks.sh: |+
    #!/bin/bash
    echo "executing extension for: $@"
    CURRENT_CHECKPOINT=$1
    if [ "$CURRENT_CHECKPOINT" == "ext_deployer_pre_exit" ] ; then
      sh my_deployer_extension.sh
    fi
  my_deployer_extension.sh: |+
    #!/bin/bash
    echo "executing my_deployer_extension"
...
```

Specify the name of your ConfigMap in the **ocbc.bc.extensions.scriptsConfigName** key in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

Since Billing Care is a web application that is deployed on WebLogic Server, refer to the WebLogic Server documentation for information about overriding timeouts, cookie attributes, and so on. See "web.xml Deployment Descriptor Elements" and "weblogic.xml Deployment Descriptor Elements" in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server* for more information about these configurations. You can find files to help you with this configuration in the **oc-cn-op-job-helm-chart/templates** directory.

About Billing Care Volume Mounts

The Billing Care container requires Kubernetes volume mounts for sharing the domain and application file system between the WebLogic Cluster servers. There is one volume for the domain and one for batch payments. By default, these are created dynamically, using the provisioner defined in BRM, in the **storage-class** key in **oc-cn-op-job-helm-chart**.

To change the volume type or provider, modify the following keys in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

- **ocbc.bc.volume.domain.createOption** for the domain file system for Billing Care.
- **ocbc.bc.volume.batchPayment.createOption** for the batch payments file system.

Creating a WebLogic Domain and Installing the Billing Care Application

The WebLogic domain is created by a Kubernetes Deployment when **oc-cn-op-job-helm-chart** is installed. The same job also installs the Billing Care application and deploys the application WAR file onto the WebLogic Cluster.

The **oc-cn-op-job-helm-chart** chart also:

- Creates a Kubernetes ConfigMap and Secrets, which are used throughout the life-cycle of the WebLogic domain.
- Initializes the **PersistentVolumeClaim** for the domain and application file system as well as third-party libraries.

Note:

The **override-values.yaml** file that you use for this chart must include BRM override values.

After you install **oc-cn-op-job-helm-chart**, wait until the Kubernetes deployment has reached the **1/1 Running** status. Then, you can install or upgrade **oc-cn-helm-chart** for Billing Care services.

After the deployment is running, don't delete the chart. Its resources will be used for starting and stopping the servers through **oc-cn-helm-chart**.

Setting Up SSO for Billing Care

SSO allows users to log in to applications using a single user name and password combination. You set up SSO for Billing Care cloud native services by using SAML 2.0.

To set up SSO for Billing Care:

1. Export the SAML 2.0 metadata XML file from your identity and access management (IAM) system.

   For example, if you are using Oracle Access Management, you can export the file by following the instructions in "Exporting Metadata" in *Oracle Fusion Middleware Administering Oracle Access Management*.

2. Add the metadata XML file to your BRM cloud native deployment by doing one of the following:

   - Rename the metadata XML file to **metadata.xml**, and then move **metadata.xml** to the **oc-cn-op-job-helm-chart/billingcare/idp** directory.
   - Pre-create the IDP metadata ConfigMap for Billing Care and set the **ocbc.bc.configEnv.extMetadataCM** key in your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

3. Configure the KeyStores needed by SAML by doing one of the following:

   - Generate the Identity and Trust KeyStores and then move your files, such as **identity.p12** and **trust.p12**, under the **oc-cn-op-job-helm-chart/billingcare/keystore** directory.
   - Pre-create the Kubernetes Secret for the Identity and Trust KeyStore files and set the **ocbc.bc.configEnv.extKeystoreSecret** key in your **override-values.yaml** file for both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**.

     For more information, see "Managing Wallet and KeyStore Certificates" in *BRM Cloud Native System Administrator's Guide*.

4. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, set the following keys:

   - **ocbc.bc.configEnv.isSSOEnabled**: Set this to **true**.
   - **ocbc.bc.configEnv.keystoreAlias**: Set this to the private key alias of the KeyStore.
   - **ocbc.bc.configEnv.keystoreType**: Set this to the file type of the SSL Identity and Trust store, which is either **PKCS12** or **JKS**. The default is **PKCS12**.

- **ocbc.bc.configEnv.keystoreIdentityFileName**: Set this to the name of the Identity KeyStore file.
- **ocbc.bc.configEnv.keystoreTrustFileName**: Set this to the name of the Trust KeyStore file.
- **ocbc.bc.configEnv.samlAsserterName**: Set this to the name of the SAML Asserter. The default is **samlBCAsserter**.
- **ocbc.bc.configEnv.ssoPublishedSiteURL**: Set this to the base URL that is used to construct endpoint URLs. This is typically the load balancer host and port at which the server is visible externally. It must be appended with **/saml2**. For example: **https://***LoadBalancerHost***:***LoadBalancerPort***/saml2**.
- **ocbc.bc.configEnv.ssoDefaultURL**: Set this to the URL where unsolicited authentication responses are sent if they do not contain an accompanying target URL.
- **ocbc.bc.secretVal.keystoreIdentityPassword**: Set this to the StorePass for the Identity KeyStore.
- **ocbc.bc.secretVal.keystoreKeyPassword**: Set this to the KeyPass for the Identity KeyStore.
- **ocbc.bc.secretVal.keystoreTrustPassword**: Set this to the StorePass for the Trust KeyStore.

5. Configure your load balancer's rules to send responses to the Billing Care WebLogic domain with **/saml2** appended to the URL path.

   Note:

   Add this rule to your existing load balancer rules for routing responses to Billing Care (**/bc**), the load balancer host name, and so on.

   See "Installing an Ingress Controller".
6. Deploy your Billing Care cloud native services by following the instructions in "Deploying BRM Cloud Native Services".
7. After Billing Care is deployed, retrieve the **sp-metadata-admin-server.xml** file from the **/shared/domains/***domainUID* directory in your container, where *domainUID* is the name of your Billing Care domain specified in the **ocbc.bc.wop.domainUID** key.

   The XML file configures the Web SSO Provider Partner. It contains the partner's KeyStore certificates, SAML assertion details, and the URLs where the SAML Identity Provider redirects to provide access to Billing Care.
8. Create a profile for your identity provider partner by loading the **sp-metadata-admin-server.xml** file into your IAM system.

   For example, if you are using Oracle Access Management, you can load the file by following the instructions in "Creating Remote Identity Provider Partners" in *Oracle Fusion Middleware Administering Oracle Access Management*.

Setting Up Local Users and Groups for Billing Care

You have the option to customize the values for **oc-cn-op-job-helm-chart** to create users and groups locally in Oracle WebLogic Server. This would be especially useful for test environments where you might not have Identity Providers or LDAPs available. The groups for the admin user for WebLogic Server cannot be modified using this procedure.

Any passwords must be encoded using Base64. You can leave the password blank, but then the user will not be able to log in to the application directly.

To set up local users and groups for Billing Care, define the keys under **ocbc.bc.wlsUserGroups** in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**. For example:

```
ocbc:
    bc:
        wlsUserGroups:
            groups:
            -   name: "GroupA"
                description: "GroupA Description"
            -   name: "GroupB"
                description: "GroupB Description"
            users:
            -   name: csr1
                description: "csr1 description"
                password: "Base64_password"
                groups:
                -   "GroupA"
                -   "GroupB"
            -   name: csr2
                description: "csr2 description"
                password: "Base64_password"
                groups:
                -   "GroupB"
```

Starting and Stopping WebLogic Servers

When you install **oc-cn-op-job-helm-chart**, the default configuration sets up a WebLogic Cluster with five Managed Servers. When you install or upgrade **oc-cn-helm-chart** for the Billing Care service, two of the Managed Servers and one Admin Server are started.

By modifying the **override-values.yaml** file for **oc-cn-helm-chart**, you can control:

- The total number of Managed Servers and the initial server start up by using the **totalManagedServers** and **initialServerCount** keys.
- Whether the servers are started or stopped by using the **serverStartPolicy** key. To start the Admin Servers and the Managed Servers in a Cluster, set the key to **IF_NEEDED**. To stop all servers, set the key to **NEVER**.

Note:

The keys in the **override-values.yaml** file should be the same as the ones used in **oc-cn-op-job-helm-chart** for keys that are common in both charts.

After you modify the **override-values.yaml** file, update the Helm release for the changes to take effect:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.

- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

Configuring the Billing Care REST API

You use the Billing Care REST API to integrate an external customer management application with BRM. This allows you to manage billing and rating in BRM and then manage your customers' accounts and bills in your external application. For more information, see *REST API Reference for Billing Care*.

To configure the Billing Care REST API to work with BRM cloud native:

1. Override the Billing Care REST API-specific keys from the **values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding Billing Care REST API Keys for oc-cn-op-job-helm-chart".
2. Override the Billing Care REST API-specific keys from the **values.yaml** file for **oc-cn-helm-chart**. See "Adding Billing Care REST API Keys for oc-cn-helm-chart".
3. Set up volume mounts for the Billing Care REST API. See "About Billing Care REST API Volume Mounts".
4. Create a WebLogic domain and install the Billing Care REST API. See "Creating a WebLogic Domain and Installing the Billing Care REST API".
5. Set up local users and groups for Billing Care REST API. See "Setting Up Local Users and Groups for Billing Care REST API".
6. Start and stop your WebLogic servers. See "Starting and Stopping WebLogic Servers".

Note:

To set up the Billing Care REST API, ensure that you successfully complete the installation of **oc-cn-op-job-helm-chart** before you install or upgrade **oc-cn-helm-chart**.

Adding Billing Care REST API Keys for oc-cn-op-job-helm-chart

Table 9-5 lists a few important keys that directly impact the Billing Care REST API. Add these keys to your **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

For the complete set of keys to personalize your Billing Care REST API deployment, see the keys with the path **ocbc.bcws** in the **oc-cn-op-job-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocbc.bcws.secretVal** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-5 Billing Care REST API Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
isEnabled	**ocbc.bcws**	Whether to deploy, configure, and start Billing Care REST API services: • **false**: Does not create the Kubernetes resources for using the Billing Care REST API. • **true**: Creates the Kubernetes resources for using the Billing Care REST API. This is the default.
imageName	**ocbc.bcws.deployment.app**	The name of the Billing Care REST API image, such as **oracle/bcws**.
imageTag	**ocbc.bcws.deployment.app**	The tag associated with the image. This is generally the release number. Prefix the value with a colon (:). For example, **:15.1.0.0.0**.
dbSSLMode	**ocbc.bcws.configEnv**	The type of connection required to connect to the database: • **TWO_WAY**: Two-way SSL authentication is required. In this case, both the client and server must authenticate each others identity. • **ONE_WAY**: One-way SSL authentication is required. In this case, the client must authenticate the server's identity. This is the default. • **NO**: SSL authentication is not required.
dbWalletType	**ocbc.bcws.configEnv**	The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**.
extDBSSLWalletSecret **extKeystoreSecret**	**ocbc.bcws.configEnv**	The names of the pre-created Kubernetes Secrets for the Billing Care REST API KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
dbWalletPassword	**ocbc.bcws.configEnv**	The password for accessing the certificates from the TrustStore and KeyStore.
rcuJdbcURL	**ocbc.bcws.configEnv**	The connection string for connecting to the database where schemas needed by Oracle Fusion Middleware products will be created, especially OPSS.
rcuDBARole	**ocbc.bcws.configEnv**	The role of the database administrator user.
rcuArgs	**ocbc.bcws.configEnv**	The additional arguments for creating the RCU.
ldapHost	**ocbc.bcws.configEnv**	The host name or IP address of the LDAP Server (for example, OUD) where users and groups will be configured for access to the Billing Care REST API.
ldapPort	**ocbc.bcws.configEnv**	The port number on which the LDAP server is listening.
ldapGroupBase	**ocbc.bcws.configEnv**	The LDAP base DN that contains groups.
ldapUserBase	**ocbc.bcws.configEnv**	The LDAP base DN that contains users.
keystoreAlias	**ocbc.bcws.configEnv**	The private key alias of the KeyStore.
keystoreType	**ocbc.bcws.configEnv**	The file type of SSL Identity and Trust store, either **PKCS12** or **JKS**.

Table 9-5 (Cont.) Billing Care REST API Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
keystoreIdentityFileName	**ocbc.bcws.configEnv**	The file name of the Identity KeyStore.
keystoreTrustFileName	**ocbc.bcws.configEnv**	The file name of the Trust KeyStore.
tlsVersions	**ocbc.bcws.configEnv**	The list of TLS versions to support for connection with the WebLogic domain. List the version numbers in order, from lowest to highest, separated by a comma. For example: **TLSv1.2, TLSv1.3**.
reloadVersion	**ocbc.bcws.configEnv**	Update this value with any value different from the current value to force a restart of the deployer.
adminPassword	**ocbc.bcws.secretVal**	The password of the WebLogic domain's administrative user, which is used for accessing the WebLogic Console for administrative operations.
ldapPassword	**ocbc.bcws.secretVal**	The password of the LDAP Server admin user.
rcuSysDBAPassword	**ocbc.bcws.secretVal**	The password for the rcuJdbcURL database administrator.
rcuSchemaPassword	**ocbc.bcws.secretVal**	The passwords for the schemas of Oracle Fusion Middleware products that will be created by RCU, which is used by OPSS.
dbWalletPassword	**ocbc.bcws.secretVal**	The password for accessing the certificates from the TrustStore and KeyStore.
keystoreIdentityPassword	**ocbc.bcws.secretVal**	The storepass of the Identity KeyStore.
keystoreKeyPassword	**ocbc.bcws.secretVal**	The KeyPass of the Identity KeyStore.
keystoreTrustPassword	**ocbc.bcws.secretVal**	The storepass of Trust KeyStore.
domainUID	**ocbc.bcws.wop**	The name of the domain. The default is **bcws-domain**.
adminChannelPort	**ocbc.bcws.wop**	The NodePort where the admin-server's HTTP service is accessible. The default is **30711**.
serverStartPolicy	**ocbc.bcws.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count.
domain.*	**ocbc.bcws.volume**	Details about the PVC for the domain file system: • **storage**: The storage size of the volume. • **createOption**: By default, the bcws pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 9-5 (Cont.) Billing Care REST API Keys for oc-cn-op-job-helm-chart

Key	Path in values.yaml File	Description
batchPayment.*	**ocbc.bcws.volume**	Details about the PVC for the batch payment files: • **storage**: The storage size of the volume. • **createOption**: By default, the bcws pod uses dynamic volume provisioning. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocbc.bcws**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocbc.bcws**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.

Adding Billing Care REST API Keys for oc-cn-helm-chart

Table 9-6 lists a few important keys that directly impact the Billing Care REST API. Add these keys to your **override-values.yaml** file for **oc-cn-helm-chart**.

For the complete set of keys to personalize your Billing Care REST API deployment, see the keys with the path **ocbc.bcws** in the **oc-cn-helm-chart/values.yaml** file.

Caution:

Keys with the path **ocbc.bcws.secretVal** hold sensitive data. Handle them carefully with controlled access to the override file containing their values. Encode all of these values in Base64 format. See "Secrets" in *Kubernetes Concepts*.

Table 9-6 Billing Care REST API Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
appLogLevel	**ocbc**	The logging level at which application logs must be captured in log files: **SEVERE**, **WARNING**, **INFO**, **CONFIG**, **FINE**, **FINER**, **FINEST**, and **ALL**.
isEnabled	**ocbc.bcws**	Whether to deploy, configure, and start Billing Care REST API services: • **false**: Does not create the Kubernetes resources for using the Billing Care REST API. • **true**: Creates the Kubernetes resources for using the Billing Care REST API. This is the default.
imageName	**ocbc.bcws.deployment.app**	The name of the Billing Care REST API image, such as **oracle/bcws**.
imageTag	**ocbc.bcws.deployment.app**	The tag associated with the image. This is generally the release number. Prefix the value with a colon (:). For example, **:15.1.0.0.0**.

Table 9-6 (Cont.) Billing Care REST API Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
keystoreAlias	**ocbc.bcws.configEnv**	The private key alias of the KeyStore.
extDBSSLWalletSecret **extKeystoreSecret**	**ocbc.bcws.configEnv**	The names of the pre-created Kubernetes Secrets for the Billing Care REST API KeyStore certificates and wallets. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide*.
dbSSLMode	**ocbc.bcws.configEnv**	The type of connection required to connect to the database: • **TWO_WAY**: Two-way SSL authentication is required. In this case, both the client and server must authenticate each others identity. • **ONE_WAY**: One-way SSL authentication is required. In this case, the client must authenticate the server's identity. This is the default. • **NO**: SSL authentication is not required.
dbWalletType	**ocbc.bcws.configEnv**	The type of TrustStore and KeyStore file that is used for the SSL connection: **SSO** or **PKCS12**.
user.*	**ocbc.bcws.infranet**	The permissions for acessing BRM: • **login**: The username of the service that has permission to access BRM. The default is **bc_client.0.0.0.1**. • **serviceType**: The POID type of the service that has permission to access BRM. The default is **/service/admin_client**. • **serviceID**: The POID ID of the service that has permission to access BRM. The default is **416**.
connectionpool.*	**ocbc.bcws.infranet**	The size of the connection pool • **minSize**: The minimum number of connections in the pool. The default is **25**. • **maxSize**: The maximum number of connections in the pool. The default is **50**.
loglevel	**ocbc.bcws.infranet**	The log level for the **Infranet.properties** file.
idp.*	**ocbc.bcws.infranet**	The details for the Identity Provider (IdP) for authenticating clients to access Billing Care REST API: • **vendor**: The IdP application: **OAM** or **IDCS**. • **url**: The base URL of the IdP server, such as https://host:port. • **resourceServerAndScope**: The resource server and scope created in your IDP for Billing Care REST API. • **identityDomain**: The name of the Identity Provider. This is mandatory if the vendor is OAM. • **clientId**: The client ID for accessing the IdP when validating the access token. This is mandatory if the vendor is IDCS.
addOnProperties	**ocbc.bcws.infranet**	This key is empty by default. You can use this key to specify custom **Infranet.properties** values.

Table 9-6 (Cont.) Billing Care REST API Keys for oc-cn-helm-chart

Key	Path in values.yaml File	Description
secretVal.*	**ocbc.bcws**	The passwords: • **walletPassword**: The password for the wallet storing sensitive data for connecting to BRM. • **bipPassword**: The password for your BI Publisher instance. • **keystoreTrustPassword**: The StorePass for the Trust KeyStore. • **infranet.idp.clientSecret**: The client secret for communicating with the authorization server.
domainUID	**ocbc.bcws.wop**	The name of the domain. The default is **bcws-domain**.
adminChannelPort	**ocbc.bcws.wop**	The NodePort where the admin-server's HTTP service will be accessible. By default, this key is blank. **Note**: Set this key only if you want the **bcws-domain-admin-server-ext** service to deploy as NodePort.
serverStartPolicy	**ocbc.bcws.wop**	The WebLogic servers that the Operator starts when it discovers the domain: • **NEVER**: Does not start any server in the domain. • **ADMIN_ONLY**: Starts only the administration server (no managed servers will be started). • **IF_NEEDED**: Starts the administration server and clustered servers up to the replica count. This is the default.
isEnabled	**ocbc.bcws.monitoring**	Whether to enable monitoring of Billing Care REST API. See "Monitoring and Autoscaling Billing Care Cloud Native" in *BRM Cloud Native System Administrator's Guide*.
nodeSelector	**ocbc.bcws**	The node selector rules for scheduling WebLogic Server pods on particular nodes using simple selectors.
affinity	**ocbc.bcws**	The affinity rules for scheduling WebLogic Server pods on particular nodes using more powerful selectors.

Updating Infranet Properties for the Billing Care REST API

The **Infranet.properties** file entries are located in the **values.yaml** file. This makes it easier to update them.

Following is a sample configuration block (located in the **ocbc.bcws** path in **oc-cn-helm-chart**) for the **Infranet.properties** entries:

```
infranet:
    user:
        login: 'bc_client.0.0.0.1'
        serviceType: '/service/admin_client'
        serviceId: 416
    connectionpool:
        minSize: 25
        maxSize: 50
    logLevel: 3
    idp:
        vendor: IDCS
        url: https://host:port
        resourceServerAndScope:
        identityDomain:
        clientId:
    addOnProperties: ""
```

If you have custom field classes, they should be provided through the SDK **.war** file and defined here using the **addOnProperties** key. For example:

```
addOnProperties:|-
    infranet.custom.field.package=com.portal.custom
    infranet.custom.field.100011=PIN_FLD_ABC
```

To update any of these properties after an install or upgrade, update the values in **override-values.yaml** file for **oc-cn-helm-chart**. If this is an upgrade, also update the **ocbc.bcws.wop.restartVersion** key in the same file. This will force a pod restart and the new values will be used.

Adding Custom Configuration to Deployment Workflow for Billing Care REST API

You can provide additional configuration to be applied at particular checkpoints in the Billing Care REST API deployment workflow. These checkpoints are:

- **ext_deployer_pre_exit**: Called after the standard configuration in **deployer.sh** in **oc-cn-op-job-helm-chart**
- **ext_init_app_pre_exit**: Called after the standard configuration in the init-app **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**
- **ext_init_config_pre_exit**: Called after the standard configuration in the init-config **initContainer** container in both **oc-cn-op-job-helm-chart** and **oc-cn-helm-chart**

Create a ConfigMap with your configuration scripts, including a shell script named **run_hooks.sh** that calls your other scripts. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ext-scripts
data:
  run_hooks.sh: |+
    #!/bin/bash
    echo "executing extension for: $@"
    CURRENT_CHECKPOINT=$1
    if [ "$CURRENT_CHECKPOINT" == "ext_deployer_pre_exit" ] ; then
      sh my_deployer_extension.sh
    fi
  my_deployer_extension.sh: |+
    #!/bin/bash
    echo "executing my_deployer_extension"
...
```

Specify the name of your ConfigMap in the **ocbc.bcws.extensions.scriptsConfigName** key in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

About Billing Care REST API Volume Mounts

The Billing Care REST API container requires Kubernetes volume mounts for sharing the domain and application file system between the WebLogic Cluster servers. There is one volume for the domain and one for batch payments. By default, these are created dynamically, using the provisioner defined in BRM, in the **storage-class** key in **oc-cn-op-job-helm-chart**.

> **Note:**
>
> The selected location must be accessible on all worker nodes across which WebLogic Servers will be distributed based on defined nodeSelector or affinity rules.

To change the volume type or provider, modify the following keys in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

- **ocbc.bcws.volume.domain.createOption** for the domain file system for Billing Care.
- **ocbc.bcws.volume.batchPayment.createOption** for the batch payments file system.

Creating a WebLogic Domain and Installing the Billing Care REST API

The WebLogic domain is created by a Kubernetes Deployment when **oc-cn-op-job-helm-chart** is installed. The same job also installs the Billing Care REST API and deploys the application WAR file onto the WebLogic Cluster.

The **oc-cn-op-job-helm-chart** chart also:

- Creates a Kubernetes ConfigMap and Secrets, which are used throughout the life-cycle of the WebLogic domain.
- Initializes the **PersistentVolumeClaim** for the domain and application file system as well as third-party libraries.

> **Note:**
>
> The **override-values.yaml** file that you use for this chart must include BRM override values.

After you install **oc-cn-op-job-helm-chart**, wait until the Kubernetes deployment has reached the **1/1 Running** status. Then, you can install or upgrade **oc-cn-helm-chart** for Billing Care REST API services.

After the deployment is running, don't delete the chart. Its resources will be used for starting and stopping the servers through **oc-cn-helm-chart**.

Setting Up Local Users and Groups for Billing Care REST API

You have the option to customize the values for **oc-cn-op-job-helm-chart** to create users and groups locally in Oracle WebLogic Server. This would be especially useful for test environments where you might not have Identity Providers or LDAPs available. The groups for the admin user for WebLogic Server cannot be modified using this procedure.

Any passwords must be encoded using Base64. You can leave the password blank, but then the user will not be able to log in to the application directly.

To set up local users and groups for Billing Care, define the keys under **ocbc.bcws.wlsUserGroups** in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**. For example:

```
ocbc:
    bcws:
        wlsUserGroups:
            groups:
            -   name: "GroupA"
                description: "GroupA Description"
            -   name: "GroupB"
                description: "GroupB Description"
            users:
            -   name: csr1
                description: "csr1 description"
                password: "Base64_password"
                groups:
                -   "GroupA"
                -   "GroupB"
            -   name: csr2
                description: "csr2 description"
                password: "Base64_password"
                groups:
                -   "GroupB"
```

Starting and Stopping WebLogic Servers

When you install **oc-cn-op-job-helm-chart**, the default configuration sets up a WebLogic Cluster with five Managed Servers. When you install or upgrade **oc-cn-helm-chart** for the Billing Care REST API service, two of the Managed Servers and one Admin Server are started.

By modifying the **override-values.yaml** file for **oc-cn-helm-chart**, you can control:

- The total number of Managed Servers and the initial server start up by using the **totalManagedServers** and **initialServerCount** keys.
- Whether the servers are started or stopped by using the **serverStartPolicy** key. To start the Admin Servers and the Managed Servers in a Cluster, set the key to **IF_NEEDED**. To stop all servers, set the key to **NEVER**.

Note:

The keys in the **override-values.yaml** file should be the same as the ones used in **oc-cn-op-job-helm-chart** for keys that are common in both charts.

After you modify the **override-values.yaml** file, update the Helm release for the changes to take effect:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.

- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

10 Configuring ECE Services

Learn how to configure Oracle Communications Elastic Charging Engine (ECE) services by configuring and deploying the ECE Helm chart.

Topics in this document:

For information about performing administrative tasks on your ECE cloud native services, see "Administering ECE Cloud Native Services" in *BRM Cloud Native System Administrator's Guide*.

Before installing the ECE Helm chart, you must first publish the metadata, config, and pricing data from the PDC pod.

Note:

Kubernetes looks for the CPU limit setting for pods. If it's not set, Kubernetes allocates a default value of 1 CPU per pod, which causes CPU overhead and Coherence scalability issues. To prevent this from happening, override each ECE pod's CPU limit to be the maximum CPU available on the node.

Adding Elastic Charging Engine Keys

Table 10-1 lists the keys that directly impact ECE deployment. Add these keys to your **override-values.yaml** file for **oc-cn-ece-helm-chart**. In the table, *component-name* should be replaced with the name of the ECE component, such as **emgateway**, **radiusgateway**, **diametergateway**, **httpgateway**, and **ratedeventformatter**.

Table 10-1   Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
imagePullPolicy	**container**	The default value is **IfNotPresent**, which specifies to not pull the image if it's already present. Applicable values are **IfNotPresent** and **Always**.
containerPort	**container**	The port number that is exposed by this container.
createWalletsAsSecrets	N/A	Whether to create KeyStore certificates and wallets as Secrets during the ECE deployment process (**true**) or if they have been pre-created as Kubernetes Secrets (**false**). See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
chargingSettingManagementPath	**volume**	The location of the management folder, which contains the **charging-settings.xml**, **test-tools.xml**, and **migration-configuration.xml** files. The default is **/home/charging/opt/ECE/oceceserver/config/management**.
chargingSettingPath	**volume**	The location of the configuration folder for ECE. The default is **/home/charging/opt/ECE/oceceserver/config**.
extECESecret	**secretEnv**	The name of the external Kubernetes Secret for ECE. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
walletPassword	**secretEnv**	The string password for opening the wallet.
JMSQUEUEPASSWORD	**secretEnv**	The password for the JMS queue, which is stored under the key **jms.queue.notif.pwd** in the wallet.
RADIUSSHAREDSECRET	**secretEnv**	The RADIUS secret password, which is stored as **radius.secret.pwd** in the wallet.
BRMGATEWAYPASSWORD	**secretEnv**	The BRM Gateway password.
PDCPASSWORD	**secretEnv**	The PDC password, which is stored as **pdc.pwd** in the wallet. **Note**: This key must match the **pdcAdminUserPassword** key in the **override-values.yaml** file for **oc-cn-helm-chart**.
PDCKEYSTOREPASSWORD	**secretEnv**	The PDC KeyStore password, which is stored as **pdc.keystore.pwd** in the wallet. **Note**: This key must match the **keyStoreIdentityStorePass** key in the **override-values.yaml** file for **oc-cn-helm-chart**.
PERSISTENCEDATABASEPASSWORD	**secretEnv**	The database schema user password. This user is created using ece-persistence-job if it doesn't exist in the database.
ECEHTTPGATEWAYSERVERSSLKEYSTOREPASSWORD	**secretEnv**	The server SSL KeyStore password for the HTTP Gateway.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
BRM_SERVER_WALLET_PASSWD	**secretEnv**	The password to open the BRM server wallet.
BRM_ROOT_WALLET_PASSWD	**secretEnv**	The root wallet password of the BRM wallet.

BRMDATABASEPASSWORD	**secretEnv**	The password for the BRM database.
		If you are connecting ECE to a BRM multischema database, use these entries instead:
		`BRMDATABASEPASSWORD:`
		`   - schema: 1`
		`     PASSWORD: Password`
		`   - schema: 2`
		`     PASSWORD: Password`
		where:
		• **schema** is the schema number. Enter **1** for the primary schema, **2** for the secondary schema, and so on.
		• **PASSWORD** is the schema password.

SSLENABLED	**sslconnectioncertificates**	Whether SSL is enabled in ECE (**true**) or not (**false**).
DNAME	**sslconnectioncertificates**	The domain name. For example: "CN=Admin, OU=Oracle Communication Application, O=Oracle Corporation, L=Redwood Shores, S=California, C=US"
SSLKEYSTOREVALIDITY	**sslconnectioncertificates**	The validity of the KeyStore, in days. A value of **200** indicates that the validity is 200 days.
runjob	**job.sdk**	Whether the SDK job needs to be run as part of the deployment (**true**) or not (**false**). The default value is **false**. If set to **true**, a default SDK job is run as part of the Helm installation or upgrade.
schemaLoader.resources	**job**	The minimum and maximum CPU and memory resources for ece-persistence-job and ece-persistence-upgrade-job. See "Setting Minimum and Maximum CPU and Memory Values" in *BRM Cloud Native System Administrator's Guide*.
serviceFqdn	**emgateway**	The default is **ece-emg**.
tlsVersion	**customerUpdater.customerUpdaterList.oracleQueueConnectionConfiguration**	The TLS version to support, such as 1.2 or 1.3
extBRMDBSSLWalletSecret	**customerUpdater.customerUpdaterList.oracleQueueConnectionConfiguration.extBRMDBSSLWalletSecret**	The name of the external Kubernetes Secret containing the SSL TrustStore. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
replicas	*component-name*.*component-name***List**	The number of replicas to be created while deploying the chart. The default replica count is 3 for ecs server, and 1 for all other components.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
coherenceMemberName	*component-name.component-name***List**	The Coherence member name under which this component will be added to the Coherence cluster.
jmxEnabled	*component-name.component-name***List**	Whether the component is JMX-enabled (**true**) or not (**false**).
coherencePort	*component-name.component-name***List**	The optional value indicating the Coherence port used by the component.
jvmGCOpts	*component-name.component-name***List**	This field helps to provide the Java JVM options such as GC details, max memory, and min memory.
jvmJMXOpts	*component-name.component-name***List**	This field helps to provide the JMX-related option.
jvmCoherenceOpts	*component-name.component-name***List**	This field helps to provide the Coherence-related options such as the override file and cache config file.
jvmOpts	*component-name.component-name***List**	This field is empty by default, and any additional JVM arguments can be provided here.
extServerSSLKeyStoreSecret **extHttpIdentityKeystoreSecret** **extHttpTruststoreSecret** **extNrfPublicKeyLocationSecret**	**httpgateway**	The names of the external Kubernetes Secrets for the HTTP Gateway. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
labels	**charging**	The label for all pods in the deployment. The default value is **ece**.
jmxport	**charging**	The JMX port exposed by ece, which can be used to log in to JConsole. The default is **31022**.
terminationGracePeriodSeconds	**charging**	Used for graceful shutdown of the pods. The default value is 180 seconds.
persistenceEnabled	**charging**	Whether to persist the ECE cache data into the Oracle database. The default is **true**. See "Enabling Persistence in ECE" in *BRM Cloud Native System Administrator's Guide* for more information.
hpaEnabled	**charging**	Whether to enable autoscaling using Kubernetes Horizontal Pod Autoscaler. See "Setting Up Autoscaling of ECE Pods" in *BRM Cloud Native System Administrator's Guide* for more information.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
timeoutSurvivorQuorum	**charging**	The minimum number of cluster members that must remain in the cluster when the cluster service is terminating suspect members, without data loss. The default is **3**. To calculate the minimum number, use this formula: (**chargingServerWorkerNodes** – 1) * (sum of all ecs pods/**chargingServerWorkerNodes**)
chargingServerWorkerNodes	**charging**	The number of charging server worker nodes. The default is **3**.
primary.*	**charging.cluster**	The details about your primary cluster: • **clusterName**: The name of the primary cluster. • **eceServiceName**: The ECE service name that creates the Kubernetes cluster with all of the ECE components in the primary cluster. The default is **ece-server**. • **eceServicefqdnOrExternalIP**: The fully qualified domain name (FQDN) or external IP address of the ECE service running in the primary cluster. For example: **ece-server.***NameSpace***.svc.cluster.local**.
secondary.*	**charging.cluster**	The details about your secondary cluster: • **clusterName**: The name of the secondary cluster. • **eceServiceName**: The ECE service name that creates the Kubernetes cluster with all of the ECE components in the secondary cluster. The default is **ece-server**. • **eceServicefqdnOrExternalIP**: The fully qualified domain name (FQDN) or external IP address of the ECE service running in the secondary cluster. For example: **ece-server.***NameSpace***.svc.cluster.local**.
extPersistenceDBSSLWalletSecret	**charging.server.connectionConfigurations.OraclePersistenceConnectionConfigurations**	The name of the external Kubernetes Secret containing the ECE database SSL TrustStore. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
extKafkaTrustStoreSecret	**charging.kafkaConfigurations.kafkaConfigurationList**	The name of the external Kubernetes Secret containing the server Kafka SSL KeyStore. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
<tags>	**migration**	The different tags indicating the values that will be stored under **migration-configuration.xml**. The tag names are the same as the ones used in the **migration-configuration.xml** file for ease of mapping.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
extPDCKeyStoreSecret	**migration.loader.pricingUpdater**	The name of the external Kubernetes Secret containing the server PDC SSL KeyStore. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
<tags>	**testtools**	The different tags indicating the values that will be stored under **test-tools.xml**. The tag names are the same as the ones used in the **test-tools.xml** file for ease of mapping.
<module>	**log4j2.logger**	The different log levels for each module represents the logging level for the corresponding module.
<tags>	**eceproperties**	The different tags indicating the values that will be stored under **ece.properties**. The tag names are the same as the ones used in the **ece.properties** file for ease of mapping.
NotificationQueue.extJMSKeyStoreSecret **BRMGatewayNotificationQueue.extJMSKeyStoreSecret** **DiameterGatewayNotificationQueue.extJMSKeyStoreSecret**	**JMSConfiguration**	The names of the external Kubernetes Secrets for the gateway queues. See "About Using External Kubernetes Secrets" in *BRM Cloud Native System Administrator's Guide* for more information.
<tags>	**JMSConfiguration**	The different tags indicating the values that will be stored under **JMSConfiguration.xml**. The tag names are the same as the ones used in the **JMSConfiguration.xml** file for ease of mapping.
name	**secretEnv**	The user-defined name to give for the Secrets. The default is **secret-env**.
SSLENABLED	**sslconnectioncertificates**	Whether to install ECE under SSL mode (**true**) or not (**false**). The default is **true**.
external.*	**pv**	The details about the external persistent volume (PV). • **name**: The name of the external PV. The default is **external-pv**. • **createOption**: By default, the PV uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*. • **accessModes**: The access mode for the PV. The default is **ReadWriteMany**. • **capacity**: The maximum capacity of the external PV. The default is **500Mi**.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
logs.*	**pvc**	The details about the persistent volume claim (PVC) for log files. • **name**: The name for the ECE log files. The default is **logs-pv**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**. • **createOption**: By default, the PVC uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
brmconfig.*	**pvc**	The details about the PVC for BRM configuration files. • **name**: The name of the BRM Config PVC, in which all BRM configuration files such as the payload configuration file are exposed outside of the pod. The default is **brmconfig-pvc**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**. • **createOption**: By default, the PVC uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
sdk.*	**pvc**	The details about the PVC for the SDK files. • **name**: The name for the SDK PVC, in which all of the SDK files such as the configuration, sample script, and source files are exposed to the user. The default is **sdk-pvc**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**. • **createOption**: By default, the PVC uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.

Table 10-1 (Cont.) Elastic Charging Engine Keys

Key	Path in values.yaml File	Description
cdrformatter.*	**pvc**	The details about the PVC for the CDR formatter files. • **name**: The name for the CDR formatter PVC. The default is **cdrformatter-pvc**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**. • **createOption**: By default, the PVC uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
wallet.*	**pvc**	The details about the PVC for wallet files. • **name**: The name for the wallet PVC, in which the wallet directory will be stored and shared by all of the ecs pods. The default is **ece-wallet-pvc**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**. • **createOption**: By default, the PVC uses dynamic volumes. To use a static volume instead, you must add the **createOption** key. See "Using Static Volumes" in *BRM Cloud Native System Administrator's Guide*.
external.*	**pvc**	The details about the PVC for external files. • **name**: The name for the external PVC. The default is **external-pvc**. • **accessModes**: The access mode for the PVC. The default is **ReadWriteMany**. • **storage**: The storage space required initially to create this PVC. If the storage specified here is not available in the machine, ensure that the PVC is not created and that the pods do not get initialized. The default is **500Mi**.
name	**storageClass**	The name of the storage class for dynamic volume provisioning.

Enabling SSL in Elastic Charging Engine

> **Note:**
>
> For more information about securing communications between ECE and external applications, see "Securing ECE Communication" in *BRM Cloud Native System Administrator's Guide*.

To complete the configuration for SSL setup in ECE:

1. Configure the SSL KeyStores by doing one of the following:
   - Generate the Identity and Trust KeyStores and then move your files, such as **identity.p12** and **trust.p12**, under the **oc-cn-ece-helm-chart/ece/keystore** directory.
   - Pre-create the Kubernetes Secret for the Identity and Trust KeyStore files and set the **secretEnv.extECESecret** key in your **override-values.yaml** file for both **oc-cn-ece-helm-chart**.

     For more information, see "Managing KeyStore Certificates and Wallets" in *BRM Cloud Native System Administrator's Guide*.
2. Set these keys in the **override-values.yaml** file for **oc-cn-ece-helm-chart**:
   - **sslconnectioncertificates.SSLENABLED**: Set this to **true**.
   - **sslEnabled**: Set this to **true** in **emGatewayConfigurations**, **httpGatewayConfigurations**, and **BRMConnectionConfiguration**.
   - **migration.pricingUpdater.keyStoreLocation**: Set this to **/home/charging/opt/ECE/oceceserver/config/client.jks**.
   - **charging.brmWalletServerLocation**: Set this to **/home/charging/wallet/brmwallet/server/cwallet.sso**.
   - **charging.brmWalletClientLocation**: Set this to **/home/charging/wallet/brmwallet/client/cwallet.sso**.
   - **charging.brmWalletLocation**: Set this to **/home/charging/wallet/brmwallet**.
   - **charging.emGatewayConfigurations.emGatewayConfigurationList.emGateway1Config.wallet**: Set this to the BRM wallet location.
   - **charging.emGatewayConfigurations.emGatewayConfigurationList.emGateway2Config.wallet**: Set this to the BRM wallet location.
   - **charging.radiusGatewayConfigurations.wallet**: Set this to the BRM wallet location.
   - **charging.connectionConfigurations.BRMConnectionConfiguration.brmwallet**: Set this to the BRM wallet location.
3. Deploy your ECE cloud native services by following the instructions in "Deploying BRM Cloud Native Services".

Connecting ECE Cloud Native to an SSL-Enabled Database

The steps for connecting ECE cloud native to an SSL-enabled database depends on where you save your SSL certificates for various components, such as the Kafka DM, WebLogic

Server, PDC, and the persistence database. The certificates can be stored in the following locations:

- The ECE Helm chart. In this case, the ECE Helm chart creates the certificates as Kubernetes Secrets during the deployment process.
- The ECE Persistent Volumes (PVs)

You specify where the SSL certificates are located by using the **createWalletsAsSecrets** key in your **override-values.yaml** file for **oc-cn-ece-helm-chart**. Set the key to **true** if your SSL certificates are in the ECE Helm chart, and to **false** if they are in PVs.

To connect your ECE cloud native services to an SSL-enabled Oracle database:

1. If your SSL certificates are in the ECE PVs (**createWalletsAsSecrets** is **false**), do the following:
   a. Prepare for persistence schema creation.
      i. Go to the **oc-cn-ece-helm-chart** directory, and create a directory named **secrets/db_wallets/ece_ssl_db_wallet/schema1**.
      ii. Save the ECE SSL database wallet to the **schema1** directory.
      iii. From the **oc-cn-ece-helm-chart** directory, grant the necessary permissions to the **secrets** directory:

      ```
      chmod -R 775 secrets
      ```

      iv. (For multischema systems only) Create a directory named **schema2** in the **ece_ssl_db_wallet** directory and copy the ECE SSL database wallet to the **schema2** directory.
   b. Configure the SSL database wallets in the external volume mount.
      i. Go to the external volume mount location (**external-pvc**).
      ii. Create a directory named **ece_ssl_db_wallet/schema1**.
      iii. Save the ECE SSL database wallet to the **ece_ssl_db_wallet/schema1** directory.
      iv. From the external volume mount location, create a directory named **brm_ssl_db_wallet/schema1**.
      v. Save the BRM SSL database wallet to the **brm_ssl_db_wallet/schema1** directory.
      vi. From the external volume mount location, grant the necessary permissions to both new directories:

      ```
      chmod -R 775 ece_ssl_db_wallet brm_ssl_db_wallet
      ```

      vii. (For multischema systems only) Create a **schema2** directory inside both the **ece_ssl_db_wallet** and **brm_ssl_db_wallet** directories. Then, copy the ECE SSL certificates to the **ece_ssl_db_wallet/schema2** directory, and the BRM SSL certificates to the **brm_ssl_db_wallet/schema2** directory.
2. If your SSL certificates are in the ECE Helm chart (**createWalletsAsSecrets** is **true**), do the following:
   a. Place your ECE certificate files in the appropriate **oc-cn-ece-helm-chart/secrets/db_wallets/ece_ssl_db_wallet/schema*n*** directory, where *n* is the schema number.
   b. Place your BRM certificate files in the appropriate **oc-cn-ece-helm-chart/secrets/db_wallets/brm_ssl_db_wallet/schema*n*** directory, where *n* is the schema number.
3. Configure ECE for an SSL-enabled Oracle persistence database.

Under the **charging.connectionConfigurations.OraclePersistenceConnectionConfigurations** section, set the following keys:

- **dbSSLEnabled**: Set this to **true**.
- **dbSSLType**: Set this to the type of SSL connection required for connecting to the database: **oneway**, **twoway**, or **none**.
- **sslServerCertDN**: Set this to the SSL server certificate distinguished name (DN). The default is **DC=local,DC=oracle,CN=pindb**.
- **trustStoreLocation**:
  a. If **createWalletsAsSecrets** is **false**, set this to **/home/charging/ext/ece_ssl_db_wallet/schema1/cwallet.sso**.
  b. If **createWalletsAsSecrets** is **true**, set this to certificate file name.
- **trustStoreType**: Set this to the TrustStore file type: **SSO** or **PKCS12**.

4. Configure customerUpdater for an SSL-enabled Oracle AQ database queue.

   Under the **customerUpdater.customerUpdaterList.oracleQueueConnectionConfiguration** section, set the following keys:

   - **dbSSLEnabled**: Set this to **true**.
   - **dbSSLType**: Set this to the type of SSL connection required for connecting to the database: **oneway**, **twoway**, or **none**.
   - **sslServerCertDN**: Set this to the SSL server certificate distinguished name (DN). The default is **DC=local,DC=oracle,CN=pindb**.
   - **trustStoreLocation**:
     a. If **createWalletsAsSecrets** is **false**, set this to **/home/charging/ext/brm_ssl_db_wallet/schema1/cwallet.sso**.
     b. If **createWalletsAsSecrets** is **true**, set this to *certificate_file_name*.
   - **trustStoreType**: Set this to the TrustStore file type: **SSO** or **PKCS12**.

   Note:

   For database connectivity, ECE supports only the database service name and not the database service ID. Therefore, set the following keys to the database service name:

   - **charging.connectionConfigurations.OraclePersistenceConnectionConfigurations.sid**
   - **customerUpdater.customerUpdaterList.oracleQueueConnectionConfiguration.sid**

5. Configure your Oracle database to connect to the SSL-enabled BRM and ECE databases:
   a. Copy the **tnsnames.ora** and **sqlnet.ora** files from your SSL database host to your ECE cloud native instance.
   b. On your ECE cloud native instance, go to the ECE Helm chart directory.
   c. Create the **ora_files/ece** and **ora_files/brm** directories.

d. Copy the ECE database **tnsnames.ora** and **sqlnet.ora** files to the **oc-cn-ece-helm-chart/ora_files/ece/** directory.

e. In the **oc-cn-ece-helm-chart/ora_files/ece/sqlnet.ora** file, set the wallet location to **/home/charging/opt/ECE/oceceserver/config/ece_ssl_db_wallet/schema1**:

```
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA =
       (DIRECTORY = /home/charging/opt/ECE/oceceserver/config/ece_ssl_db_wallet/
schema1)
       )
   )
```

f. Copy the BRM database **tnsnames.ora** and **sqlnet.ora** files to the **oc-cn-ece-helm-chart/ora_files/brm** directory.

g. In the **oc-cn-ece-helm-chart/ora_files/brm/sqlnet.ora** file, set the wallet location to **/home/charging/opt/ECE/oceceserver/config/brm_ssl_db_wallet/schema1**:

```
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA =
       (DIRECTORY = /home/charging/opt/ECE/oceceserver/config/brm_ssl_db_wallet/
schema1)
       )
   )
```

h. From the ECE Helm chart directory, grant the permissions for the **ora_files** directory:

```
chmod -R 775 ora_files/
```

i. Copy the **ora_files** directory to the external PV mount location.

6. If createWalletsAsSecrets is true, do the following in the ECE Cloud Native environment:

   a. Place the SSL Certificates in the following directory structure inside the Helm chart:

      - **secrets/jms/brmgateway/**
      - **secrets/jms/diametergateway/**
      - **secrets/jms/ecs/**
      - **secrets/kafka/**
      - **secrets/pdc/**

   b. If the SSL certificates are deployed as external secrets, provide the respective secret names and certificate keys in your **values.yaml** file or in an override file:

```
secretEnv:
  extECESecret: external_secret_name
customerUpdater:
  customerUpdaterList:
    - oracleQueueConnectionConfiguration:
        trustStoreLocation: certificate_key
        extBRMDBSSLWalletSecret: external_secret_name
charging:
  connectionConfigurations:
    oraclePersistenceConnectionConfigurations:
      - trustStoreLocation: certificate_key
        extPersistenceDBSSLWalletSecret: external_secret_name
```

```
charging:
  kafkaConfigurations:
    kafkaConfigurationList:
      - kafkaTrustStoreLocation: certificate_key
        extKafkaTrustStoreSecret: external_secret_name
migration:
  pricingUpdater:
    keyStoreLocation: certificate_key
    extPDCKeyStoreSecret: external_secret_name
JMSConfiguration:
  NotificationQueue:
    - KeyStoreLocation: certificate_key
      extJMSKeyStoreSecret: external_secret_name
  BRMGatewayNotificationQueue:
    - KeyStoreLocation: certificate_key
      extJMSKeyStoreSecret: external_secret_name
  DiameterGatewayNotificationQueue:
    - KeyStoreLocation: certificate_key
      extJMSKeyStoreSecret: external_secret_name
```

For more information on this configuration for the HTTP Gateway, see "Adding Elastic Charging Engine Keys".

About Elastic Charging Engine Volume Mounts

Note:

You must use a provisioner that has ReadWriteMany access and sharing between pods.

The ECE container requires Kubernetes volume mounts for third-party libraries. The third-party volume mount shares the third-party libraries required by ECE from the host system with the container file system. For the list of third-party libraries to download, see *BRM Compatibility Matrix*. Place the library files under the third-party volume mount.

The default configuration comes with a hostPath PersistentVolume. For more information, see "Configure a Pod to Use a PersistentVolume for Storage" in *Kubernetes Tasks*.

To use a different type of PersistentVolume, modify the **oc-cn-ece-helm-chart/templates/ece-pvc.yaml** file.

Loading Custom Diameter AVP

To load custom Diameter AVPs into your ECE cloud native environment:

1. Create a **diameter** directory inside **external-pvc**.
2. Move the custom AVP file, such as **dictionary_custom.xml**, to the **diameter** directory.
3. If you need to load a custom AVP after ECE is set up, restart the diametergateway pod by doing the following:
   a. Increment the **diametergateway.diametergatewayList..restartCount** key by 1.

b. Run the **helm upgrade** command to update the release.

Generating CDRs for Unrated Events

By default, the httpgateway pod sends all 5G usage requests to the ecs pod for online and offline charging.

You can configure httpgateway to convert some 5G usage requests into call detail record (CDR) files based on the charging type. You can then send the CDR files to roaming partners, a data warehousing system, or legacy billing systems for rating. For more information, see "About Generating CDRs" in *ECE Implementing Charging*.

You use the following to generate CDRs:

- httpgateway pod
- cdrgateway pod
- cdrFormatter pod
- CDR database

The cdrgateway and cdrFormatter pods can be scaled together, with one each per schema, or independently of the schemas. For more information, see "Scaling the cdrgateway and cdrFormatter Pods".

For details about the CDR format, see "CHF-CDR Format" in *ECE 5G CHF Protocol Implementation Conformance Statement*.

To set up ECE cloud native to generate CDRs:

1. Configure your httpgateway pod to do the following:
   - Generate CDRs (set **cdrGenerationEnabled** to **true**).
   - Route offline charging requests to the ecs pod for rating (set **rateOfflineCDRinRealtime** to **true**) or to the cdrgateway pod for generating CDRs (set **rateOfflineCDRinRealtime** to **false**).
   - Route online charging requests to the ecs pod for rating (set **generateCDRsForOnlineRequests** to **false**) or to the cdrgateway pod for generating CDRs (set **generateCDRsForOnlineRequests** to **true**).
2. Configure the cdrgateway pod to connect to the CDR database and do the following:
   - Generate individual CDR records for each request (set **individualCdr** to **true**) or aggregate multiple requests into a CDR record based on trigger criteria (set **individualCdr** to **false**). For information about the trigger criteria, see "About Trigger Types" in *ECE Implementing Charging*.
   - Store CDR records in an Oracle NoSQL database (set **isNoSQLConnection** to **true**) or in an Oracle database (set **isNoSQLConnection** to **false**).
3. Configure the cdrFormatter pod to do the following:
   - Retrieve batches of CDR records from the CDR database and pass them to a specified cdrFormatter plug-in for processing.
   - Purge processed CDR records from the CDR database older than a specified number of days (configured in **retainDuration**).
   - Purge orphan CDR records from the CDR database.

Orphan CDR records are incomplete ones that are older than a specified number of seconds (configured in **cdrOrphanRecordCleanupAgeInSec**). Orphan CDR records can be created when your ECE system goes down due to maintenance or failure.

4. Configure the cdrFormatter plug-in to do the following:
   - Write a specified number of CDR records to each CDR file (set **maxCdrCount** to the maximum number).
   - Create JSON-formatted CDR files and then store them in your file system (set **enableDiskPersistence** to **true**) or send them to your Kafka messaging service (set **enableKafkaIntegration** to **true**).

To generate CDRs in ECE cloud native, you configure the following entries in your **override-values.yaml** file. This example configures:

- httpgateway to route both online and offline charging requests to cdrgateway.
- cdrgateway to aggregate multiple requests into a CDR record and then store it in an Oracle NoSQL database.
- cdrFormatter to retrieve CDR records in batches of 2500 from the Oracle NoSQL database and then send them to the default plug-in module. Immediately after CDR records are retrieved, cdrFormatter purges them from the database. It would also purge orphan records older than 200 seconds from the database.
- The cdrFormatter plug-in to create CDR files with a maximum of 20000 CDR records and an **.out** file name extension. It would store them in your file system in the path **/home/charging/cdr_input**.

```
cdrFormatter:
 cdrFormatterList:
   - schemaNumber: "1"
     replicas: 1
     coherenceMemberName: "cdrformatter1"
     jmxEnabled: true
     jvmGCOpts: "-XX:+UnlockExperimentalVMOptions  -XX:+AlwaysPreTouch -
XX:G1RSetRegionEntries=2048 -XX:ParallelGCThreads=10 -XX:+ParallelRefProcEnabled    -
XX:MetaspaceSize=100M -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -
XX:+PrintTenuringDistribution -XX:+PrintAdaptiveSizePolicy -XX:-UseGCLogFileRotation -
XX:+UseG1GC -XX:NumberOfGCLogFiles=99"
     jvmOpts: "-Xms16g -Xmx20g -Dece.metrics.http.service.enabled=true"
     cdrFormatterConfiguration:
       name: "cdrformatter1"
       clusterName: "BRM"
       primaryInstanceName: "cdrformatter1"
       partition: "1"
       isNoSQLConnection: "true"
       noSQLConnectionName: "noSQLConnection"
       connectionName: "oraclePersistence1brm"
       threadPoolSize: "6"
       retainDuration: "0"
       ripeDuration: "60"
       checkPointInterval: "6"
       maxPersistenceCatchupTime: "0"
       pluginPath: "ece-cdrformatter.jar"
       pluginType:
"oracle.communication.brm.charging.cdr.formatterplugin.internal.SampleCdrFormatterCustomP
lugin"
       pluginName: "cdrFormatterPlugin1"
       noSQLBatchSize: "2500"
       cdrStoreFetchSize: "2500"
       cdrOrphanRecordCleanupAgeInSec:"200"
       cdrOrphanRecordCleanupSleepIntervalInSec: "200"
       enableIncompleteCdrDetection: "false"

cdrgateway:
  cdrgatewayList:
    - coherenceMemberName: "cdrgateway1"
      replicas: 6
      jmxEnabled: true
      jvmGCOpts: "-XX:+UnlockExperimentalVMOptions  -XX:+AlwaysPreTouch -
XX:G1RSetRegionEntries=2048 -XX:ParallelGCThreads=10 -XX:+ParallelRefProcEnabled    -
XX:MetaspaceSize=100M -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -
XX:+PrintTenuringDistribution -XX:+PrintAdaptiveSizePolicy -XX:-UseGCLogFileRotation -
XX:+UseG1GC -XX:NumberOfGCLogFiles=99"
      jvmJMXOpts: "-Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.authenticate=false -
Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.local.only=false"
      jvmCoherenceOpts: "-Dpof.config=charging-pof-config.xml -
Dcoherence.override=charging-coherence-override-dev.xml  -Dcoherence.security=false -
Dsecure.access.name=admin"
      jvmOpts: "-Xms6g -Xmx8g -Dece.metrics.http.service.enabled=true -
DcdrServerCorePoolSize=64 -Dserver.sockets.metrics.bind-address=0.0.0.0 -
Dece.metrics.http.port=19612"
      restartCount: "0"
      cdrGatewayConfiguration:
        name: "cdrgateway1"
        clusterName: "BRM"
        primaryInstanceName: "cdrgateway1"
        schemaNumber: "1"
        isNoSQLConnection: "true"
        noSQLConnectionName: "noSQLConnection"
        connectionName: "oraclePersistence1"
        cdrPort: "8084"
        cdrHost: "ece-cdrgatewayservice"
        individualCdr: "false"
        cdrServerCorePoolSize: "32"
        cdrServerMaxPoolSize: "256"
        enableIncompleteCdrDetection: "false"
        retransmissionDuplicateDetectionEnabled: "false"

httpgateway:
   cdrGenerationEnabled: "true"
   cdrGenerationStandaloneMode: "true"
   rateOfflineCDRinRealtime: "false"
   generateCDRsForOnlineRequests: "true"
   httpgatewayList:
      - coherenceMemberName: "httpgateway1"
        replicas: 8
        maxreplicas: 8
        jvmGCOpts: "-XX:+AlwaysPreTouch -XX:G1RSetRegionEntries=2048 -
XX:ParallelGCThreads=10 -XX:+ParallelRefProcEnabled    -XX:MetaspaceSize=100M -
XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -
XX:+PrintTenuringDistribution -XX:+PrintAdaptiveSizePolicy -XX:-UseGCLogFileRotation -
XX:+UseG1GC -XX:NumberOfGCLogFiles=99"
        jvmOpts: "-Xms10g -Xmx14g -Djava.net.preferIPv4Addresses=true -
Dece.metrics.http.service.enabled=true -Dserver.sockets.metrics.bind-address=0.0.0.0 -
Dece.metrics.http.port=19612"
        httpGatewayConfiguration:
           name: "httpgateway1"
           processingThreadPoolSize: "200"
           processingQueueSize: "32768"
           kafkaBatchSize: "10"

   connectionConfigurations:
         OraclePersistenceConnectionConfigurations:
              retryCount: "1"
              retryInterval: "1"
              maxStmtCacheSize: "100"
              connectionWaitTimeout: "3000"
              timeoutConnectionCheckInterval: "3000"
              inactiveConnectionTimeout: "3000"
              databaseConnectionTimeout: "6000"
              persistenceInitialPoolSize: "4"
              persistenceMinPoolSize: "4"
              persistenceMaxPoolSize: "20"
              reloadInitialPoolSize: "0"
              reloadMinPoolSize: "0"
              reloadMaxPoolSize: "20"
              ratedEventFormatterInitialPoolSize: "6"
              ratedEventFormatterMinPoolSize: "6"
              ratedEventFormatterMaxPoolSize: "24"

charging:
   cdrFormatterPlugins:
     cdrFormatterPluginConfigurationList:
       cdrFormatterPluginConfiguration:
         name: "cdrFormatterPlugin1"
         tempDirectoryPath: "/tmp/tmp"
         doneDirectoryPath: "/home/charging/cdr_input"
         doneFileExtension: ".out"
         enableKafkaIntegration: "false"
         enableDiskPersistence: "true"
         maxCdrCount: "20000"
         staleSessionCauseForRecordClosingString: "PARTIAL_RECORD"
         enableStaleSessionCleanupCustomField: "false"
```

Scaling the cdrgateway and cdrFormatter Pods

To increase performance and throughput, you can scale the cdrgateway and cdrFormatter pods together, with one each per schema, or scale them independently of the schemas.

Figure 10-1 shows an example of scaled cdrgateway and cdrFormatter pods that have CDR storage in an Oracle Database. This example contains:

- One cdrgateway multi-replica deployment for all ECE schemas. All cdrgateway replicas have a single CDR Gateway service acting as a front end to httpgateway.
- One cdrFormatter single-replica deployment for each ECE schema. Each cdrFormatter reads persisted CDRs from its associated ECE schema.

httpgateway forwards CDR requests to cdrgateway replicas in round-robin fashion. In this example, cdrgateway replicas 1-0, 1-1, and 1-2 persist CDRs in schema 1 tables, and replicas 1-3, 1-4, and 1-5 persist CDRs in schema 2 tables.

Figure 10-1   Scaled Architecture with an Oracle Database

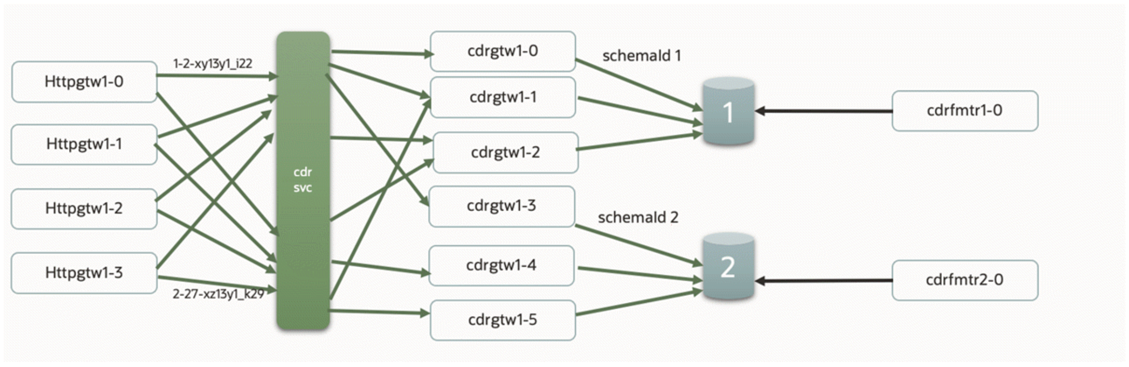
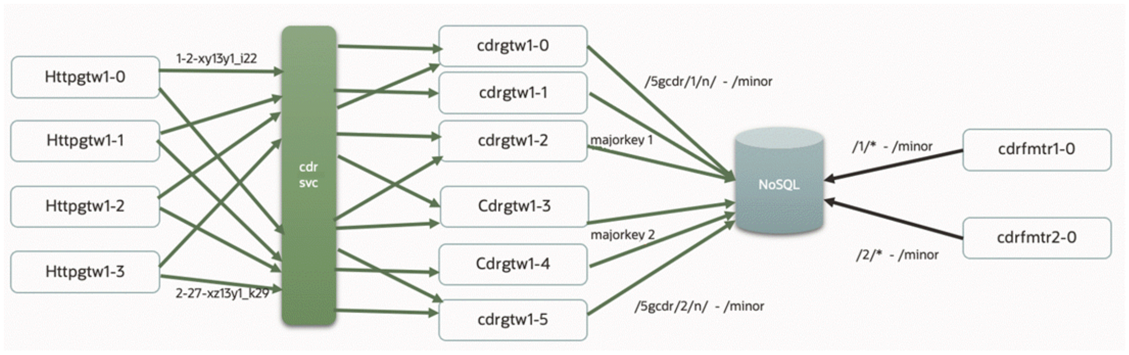


Figure 10-2 shows an example of scaled cdrgateway and cdrFormatter pods that have CDR storage in an Oracle NoSQL Database. This example contains:

- One cdrgateway multi-replica deployment for all ECE schemas. All cdrgateway replicas have a single CDR Gateway service acting as a front end to the httpgateway.
- One cdrFormatter single-replica deployment for each major key partition in the ECE schema. Each cdrFormatter reads persisted CDRs from its associated partition.

Figure 10-2   Scaled Architecture with a NoSQL Database



Configuring ECE to Support Prepaid Usage Overage

You can configure ECE cloud native to capture any overage amounts by prepaid customers during an active session, which can help you prevent revenue leakage. If the network reports that the number of used units during a session is greater than a customer's available allowance, ECE cloud native charges the customer up to the available allowance. It can then create an overage record with information about the overage amount and sends it to the ECE Overage topic. You can create a custom solution for reprocessing the overage amount later on.

For example, assume a customer has a prepaid balance of 100 minutes, but uses 130 minutes during a session. ECE cloud native would charge the customer for 100 minutes, create an overage record for the remaining 30 minutes of usage, and then write the overage topic to the ECE Overage Kafka topic.

When the prepaid usage overage is disabled, ECE cloud native charges the customer for the full amount regardless of the amount of funds in the customer's balance.

To configure ECE cloud native to support prepaid usage overage, do the following:

- Ensure that ECE cloud native is connected to your Kafka Server
- Enable ECE cloud native to support prepaid usage overage
- Create an ECE Overage topic in your Kafka Server

To do so, set the following keys in your **override-values.yaml** file for **oc-cn-ece-helm-chart** and then run the **helm upgrade** command:

- **charging.kafkaConfigurations.kafkaConfigurationList.***: Specify how to connect ECE to your Kafka Server.
- **charging.server.checkReservationOverImpact**: Set this to **true**.
- **charging.kafkaConfigurations.kafkaConfigurationList.overageTopicName**: Set this to the name of the Kafka topic where ECE will publish overage records.

Note:

If your system does not contain Kafka topics, you can configure ECE to push overage details to a separate log file instead. To do so, in your **override-values.yaml** file for **oc-cn-ece-helm-chart**, set the **charging.ecs.jvmOpts** key to **-Deceoveragelogdir=/home/charging/opt/ECE/oceceserver/logs**.

Recording Failed ECE Usage Requests

ECE cloud native may occasionally fail to process usage requests. For example, a data usage request could fail because a customer has insufficient funds. You can configure ECE cloud native to publish details about failed usage requests, such as the user ID and request payload, to the ECE failure topic in your Kafka server. Later on, you can reprocess the usage requests or view the failure details for analysis and reporting.

To configure ECE cloud native to record failed ECE usage requests:

- Ensure that ECE cloud native is connected your Kafka Server
- Enable the recording of failed ECE usage requests
- Create an ECE failure topic in your Kafka Server

To do so, set the following keys in your **override-values.yaml** file for **oc-cn-helm-chart**:

- **charging.kafkaConfigurations.kafkaConfigurationList.***: Specify how to connect ECE to your Kafka Server.
- **charging.kafkaConfigurations.kafkaConfigurationList.persistFailedRequestsToKafkaTopic**: Set this to **true**.
- **charging.kafkaConfigurations.kafkaConfigurationList.failureTopicName**: Set this to the name of the topic that stores information about failed ECE usage requests.

Loading BRM Configuration XML Files

BRM is configured by using the **pin_notify** and **payloadconfig_ece_sync.xml** files. To ensure that the BRM pod can access these files for configuring the EAI Java Server (eai_js), they are exposed through the brm_config PVC within the pricingupdater pod. When new metadata is synchronized with ECE, if there are updates to the payload configuration file, it will create a new file in the location which can be accessed and configured in BRM.

For more information, see "Enabling Real-Time Synchronization of BRM and ECE Customer Data Updates" in *ECE Implementing Charging*.

Setting Up Notification Handling in ECE

You can configure ECE cloud native to send notifications to a client application or an external application during an online charging session. For example, ECE cloud native could send a notification when a customer has breached a credit threshold or when a customer needs to request reauthorization.

You can set up ECE cloud native to send notifications by using either Apache Kafka topics or Oracle WebLogic queues:

- Creating an Apache Kafka Notification Topic
- Creating an Oracle WebLogic Notification Queue

Creating an Apache Kafka Notification Topic

To create notification topics in Apache Kafka:

1. Create these Kafka topics either in the Kafka **entrypoint.sh** script or after the Kafka pod is ready:
   - **kafka.topicName**: ECENotifications
   - **kafka.suspenseTopicName**: ECESuspenseQueue
2. In the ZooKeeper runtime ConfigMap, set the **ece-zookeeper-0.ece-zookeeper.***ECENameSpace***.svc.cluster.local** key to the name of the Kafka Cluster.
3. Set these Kafka and ZooKeeper-related environment variables appropriately:
   - **KAFKA_PORT**: Set this to the port number in which Apache Kafka is up and running.
   - **KAFKA_HOST_NAME**: Set this to the host name of the machine in which Apache Kafka is up and running. If it contains multiple Kafka brokers, create a comma-separated list.
   - **REPLICATION_FACTOR**: Set this to the number of topic replications to create.
   - **PARTITIONS**: Set this to the total number of Kafka partitions to create in your topics. The recommended number to create is calculated as follows:

     [(Max Diameter Gateways * Max Peers Per Gateway) + (1 for BRM Gateway) + Internal Notifications]
   - **TOPIC_NAME**: Set this to **ECENotifications**. This is the name of the Kafka topic where ECE will publish notifications.
   - **SUSPENSE_TOPIC_NAME**: Set this to **ECESuspenseQueue**. This is the name of the Kafka topic where BRM will publish failed notifications and will retry later.
   - **ZK_CLUSTER**: Set this to the name of your ZooKeeper cluster. This should match the value you set in step 2.
   - **ZK_CLIENT_PORT**: Set this to the port number in which ZooKeeper listens for client connections.
   - **ZK_SERVER_PORT**: Set this to the port number of the ZooKeeper server.

4. Ensure that the Kafka and ZooKeeper pods are in a READY state.
5. Set these keys in your **override-values.yaml** file for **oc-cn-ece-helm-chart**:
   - **charging.server.kafkaEnabledForNotifications**: Set this to **true**.
   - **charging.server.kafkaConfigurations.name**: Set this to the name of your ECE cluster.
   - **charging.server.kafkaConfigurations.hostname**: Set this to the host name of the machine on which Kafka is up and running.
   - **charging.server.kafkaConfigurations.topicName**: Set this to **ECENotifications**.
   - **charging.server.kafkaConfigurations.suspenseTopicName**: Set this to **ECESuspenseQueue**.
6. Install the ECE cloud native service by entering this command from the **helmcharts** directory:

   ```
   helm install EceReleaseName oc-cn-ece-helm-chart --namespace BrmNameSpace --values
   OverrideValuesFile
   ```

The notification topics are created in Apache Kafka.

Creating an Oracle WebLogic Notification Queue

To create notification queues and topics in Oracle WebLogic:

1. Ensure the following:
   - Oracle WebLogic is running in your Kubernetes cluster.
   - A separate WebLogic domain for the ECE Notification queues has been created.

     Note:

     Do not create your ECE notification queues in an existing WebLogic domain. For example, do not use the Billing Care, Business Operations Center, PDC, or Billing Care REST API domains.

   - The following third-party libraries are in the **3rdparty_jars** directory inside **external-pvc**:
     – **external-pvc**: **com.oracle.weblogic.beangen.general.api.jar**
     – **wlthint3client.jar**
   - For SSL-enabled WebLogic in a disaster recovery environment, move a common JKS certificate file for all sites to the **ece_ssl_keystore** directory inside **external-pvc**.
2. Create an **override-values.yaml** file for **oc-cn-ece-helm-chart**.
3. Set the following keys in your **override-values.yaml** file:
   - Set the **secretEnv.JMSQUEUEPASSWORD** key to the WebLogic user password.
   - If WebLogic SSL is enabled, set the **secretEnv.NOTIFYEVENTKEYPASS** key to the KeyStore password.
   - Set the **job.jmsconfig.runjob** key to **true**.
   - If the job needs to create the ECE JMS module and subdeployment, set the **job.jmsconfig.preCreateJmsServerAndModule** key to **true**.

- Set the **charging.server.weblogic.jmsmodule** key to **ECE**.
- Set the **charging.server.weblogic.subdeployment** key to **ECEQueue**.
- Set the **charging.server.kafkaEnabledForNotifications** key to **false**.
- In the **JMSConfiguration** section, set the **HostName**, **Port**, **Protocol**, **ConnectionURL**, and **KeyStoreLocation** keys to the appropriate values for your system.

For more information about these keys, see Table 10-1.

4. Copy the SSL certificate file (**client.jks**) to the **ece_ssl_keystore** directory in the external PVC.
5. Install the ECE cloud native service by entering this command from the **helmcharts** directory:

```
helm install EceReleaseName oc-cn-ece-helm-chart --namespace BrmNameSpace --values
OverrideValuesFile
```

The following are created in the ECE domain of your WebLogic Server:

- A WebLogic notification topic named **NotificationTopic**.
- A WebLogic notification queue named **SuspenseQueue**.
- A WebLogic connection factory named **NotificationFactory**.

Next, configure the connection factory resource so your clients can connect to the ECE notification queues and topics in Oracle WebLogic.

To configure the connection factory resource:

1. On the WebLogic Server in which the JMS ECE notification queue resides, sign in to WebLogic Server Administration Console.
2. In the **Domain Structure** tree, expand **Services**, expand **Messaging**, and then click **JMS Modules**.

   The Summary of JMS Modules page appears.
3. In the **JMS Modules** table, click on the name **ECE**.

   The Settings for ECE page appears.
4. In the **Summary of Resources** table, click on the name **NotificationFactory**.

   The Settings for NotificationFactory page appears.
5. Click the **Configuration** tab, and then click the **Client** tab.
6. On the **Client** page, do the following:
   a. In **Client ID Policy**, select **Unrestricted**.
   b. In **Subscription Sharing Policy**, select **Sharable**.
   c. In **Reconnect Policy**, select **None**.
   d. Click **Save**.
7. Click the **Transactions** tab.
8. On the **Transactions** page, do the following:
   a. In **Transaction Timeout**, enter **2147483647** which is the maximum timeout value.
   b. Click **Save**.

For more information, see Oracle WebLogic Administration Console Online Help.

Configuring ECE for a Multischema BRM Environment

If your BRM database contains multiple schemas, you must configure ECE to connect to each schema.

To configure ECE for a BRM multischema database:

1. Open your **override-values.yaml** file for the **oc-cn-ece-helm-chart** chart.
2. Specify the password for accessing each schema in the BRM database. To do so, configure these keys for each schema:
   - **secretEnv.BRMDATABASEPASSWORD.schema**: Set this to the schema number. Enter **1** for the primary schema, **2** for the secondary schema, and so on.
   - **secretEnv.BRMDATABASEPASSWORD.PASSWORD**: Set this to the schema password.

   This shows example settings for two schemas:

   ```
   secretEnv:
      BRMDATABASEPASSWORD:
         - schema: 1
           PASSWORD: Password
         - schema: 2
           PASSWORD: Password
   ```
3. Configure a customerUpdater pod for each schema. To do so, add a **-schemaNumber** list for each schema. In the list:
   - Set the **SchemaNumber** key to **1** for the primary schema, **2** for the secondary schema, and so on.
   - Set the **amtAckQueueName** key to the fully qualified name of the acknowledgment queue to which the **pin_amt** utility listens to Account Migration Manager (AMM)-related acknowledgment events. The value is in the format *primarySchema***.ECE_AMT_ACK_QUEUE**, where *primarySchema* is the name of the primary schema.
   - Set the **hostName** and **jdbcUrl** keys to their corresponding values for each schema.

   This shows example settings for two schemas:

   ```
   customerUpdater:
      customerUpdaterList:
         - schemaNumber: "1"
           coherenceMemberName: "customerupdater1"
           replicas: 1
           jmxEnabled: true
           coherencePort: ""
           jvmGCOpts: ""
           jvmJMXOpts: ""
           jvmCoherenceOpts: ""
           jvmOpts: ""
           jmxport: ""
           restartCount: "0"
           oracleQueueConnectionConfiguration:
              name: "customerupdater1"
              gatewayName: "customerupdater1"
              hostName: ""
              port: "1521"
              sid: "pindb"
              userName: "pin"
   ```

```
           jdbcUrl: ""
           queueName: "IFW_SYNC_QUEUE"
           suspenseQueueName: "ECE_SUSPENSE_QUEUE"
           ackQueueName: "ECE_ACK_QUEUE"
           amtAckQueueName: "pin0101.ECE_AMT_ACK_QUEUE"
           batchSize: "1"
           dbTimeout: "900"
           retryCount: "10"
           retryInterval: "60"
           walletLocation: "/home/charging/wallet/ecewallet/"

      - schemaNumber: "2"
        coherenceMemberName: "customerupdater2"
        replicas: 1
        jmxEnabled: true
        coherencePort: ""
        jvmGCOpts: ""
        jvmJMXOpts: ""
        jvmCoherenceOpts: ""
        jvmOpts: ""
        jmxport: ""
        oracleQueueConnectionConfiguration:
           name: "customerupdater2"
           gatewayName: "customerupdater2"
           hostName: ""
           port: "1521"
           sid: "pindb"
           userName: "pin"
           jdbcUrl: ""
           queueName: "IFW_SYNC_QUEUE"
           suspenseQueueName: "ECE_SUSPENSE_QUEUE"
           ackQueueName: "ECE_ACK_QUEUE"
           amtAckQueueName: "pin0101.ECE_AMT_ACK_QUEUE"
           batchSize: "1"
           dbTimeout: "900"
           retryCount: "10"
           retryInterval: "60"
           walletLocation: "/home/charging/wallet/ecewallet/"
```

4. Configure a ratedEventFormatter pod for processing rated events belonging to each BRM schema. To do so, add a **-schemaNumber** list for each schema. In the list, set the **schemaNumber** and **partition** keys to **1** for the primary schema, **2** for the secondary schema, and so on.

   This shows example settings for two schemas:

```
ratedEventFormatter:
   ratedEventFormatterList:
      - schemaNumber: "1"
        replicas: 1
        coherenceMemberName: "ratedeventformatter1"
        jmxEnabled: true
        coherencePort:
        jvmGCOpts: ""
        jvmJMXOpts: ""
        jvmCoherenceOpts: ""
        jvmOpts: ""
        jmxport: ""
        restartCount: "0"
        ratedEventFormatterConfiguration:
           name: "ratedeventformatter1"
           primaryInstanceName: "ratedeventformatter1"
           partition: "1"
```

```
                noSQLConnectionName: "noSQLConnection"
                connectionName: "oraclePersistence1"
                threadPoolSize: "6"
                retainDuration: "0"
                ripeDuration: "600"
                checkPointInterval: "6"
                maxPersistenceCatchupTime: "60"
                siteName: ""
                pluginPath: "ece-ratedeventformatter.jar"
                pluginType:
"oracle.communication.brm.charging.ratedevent.formatterplugin.internal.BrmCdrPluginDi
rect"
                pluginName: "brmCdrPlugin1"
                noSQLBatchSize: "25"

          - schemaNumber: "2"
            replicas: 1
            coherenceMemberName: "ratedeventformatter2"
            jmxEnabled: true
            coherencePort:
            jvmGCOpts: ""
            jvmJMXOpts: ""
            jvmCoherenceOpts: ""
            jvmOpts: ""
            jmxport: ""
            ratedEventFormatterConfiguration:
                name: "ratedeventformatter2"
                primaryInstanceName: "ratedeventformatter2"
                partition: "2"
                noSQLConnectionName: "noSQLConnection"
                connectionName: "oraclePersistence1"
                threadPoolSize: "6"
                retainDuration: "0"
                ripeDuration: "600"
                checkPointInterval: "6"
                maxPersistenceCatchupTime: "60"
                siteName: ""
                pluginPath: "ece-ratedeventformatter.jar"
                pluginType:
"oracle.communication.brm.charging.ratedevent.formatterplugin.internal.BrmCdrPluginDi
rect"
                pluginName: "brmCdrPlugin1"
                noSQLBatchSize: "25"
```

5. Save and close your **override-values.yaml** file for **oc-cn-ece-helm-chart**.
6. In the **oc-cn-ece-helm-chart/templates/charging-settings.yaml** ConfigMap, add **poidIdConfiguration** in **itemAssignmentConfig** for each schema.

   This shows example settings for three schemas:

```
<itemAssignmentConfigconfig-
class="oracle.communication.brm.charging.appconfiguration.beans.item.ItemAssignmentCo
nfig" itemAssignmentEnabled="true" delayToleranceIntervalInDays="0"
poidPersistenceSafeCount="12000">
   <schemaConfigurationGroup config-class="java.util.ArrayList">
      <poidIdConfigurationconfig-
class="oracle.communication.brm.charging.appconfiguration.beans.item.PoidIdConfigurat
ion" schemaName="1" poidQuantity="2000000">
      </poidIdConfiguration>
      <poidIdConfigurationconfig-
class="oracle.communication.brm.charging.appconfiguration.beans.item.PoidIdConfigurat
ion" schemaName="2" poidQuantity="2000000">
      </poidIdConfiguration>
      <poidIdConfigurationconfig-
class="oracle.communication.brm.charging.appconfiguration.beans.item.PoidIdConfigurat
ion" schemaName="3" poidQuantity="2000000">
      </poidIdConfiguration>
   </schemaConfigurationGroup>
</itemAssignmentConfig>
```

After you deploy **oc-cn-ece-helm-chart** in "Deploying BRM Cloud Native Services", the ECE pods will be connected to your BRM database schemas.

11

Deploying BRM Cloud Native Services

Learn how to deploy Oracle Communications Billing and Revenue Management (BRM) cloud native services by running the Helm install command.

Topics in this document:

Deploying BRM Cloud Native Services

Note:

The **oc-cn-init-db-helm-chart** and **oc-cn-helm-chart** charts must be deployed in different namespaces.

To deploy BRM cloud native services, do this:

1. Create a namespace for the BRM Helm chart.

   ```
   kubectl create namespace BrmNameSpace
   ```

   where *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.

2. Validate the content of your Helm charts by using the **Helm lint** command.

   - For Helm 3.6.0 and later releases, enter these commands from the **helmcharts** directory:

     ```
     helm lint --strict oc-cn-helm-chart --values oc-cn-helm-chart/values.yaml --
     values OverrideValuesFile
     helm lint --strict oc-cn-ece-helm-chart --values oc-cn-ece-helm-chart/
     values.yaml --values OverrideValuesFile
     helm lint --strict oc-cn-op-job-helm-chart --values oc-cn-op-job-helm-chart/
     values.yaml --values OverrideValuesFile
     ```

   - For previous Helm releases, enter these commands from the **helmcharts** directory:

     ```
     helm lint --strict oc-cn-helm-chart
     helm lint --strict oc-cn-ece-helm-chart
     helm lint --strict oc-cn-op-job-helm-chart
     ```

   You'll see this if the commands complete successfully:

   ```
   3 chart(s) listed, no failures
   ```

3. If you are using Pricing Design Center (PDC), Billing Care, the Billing Care REST API, Web Services Manager on WebLogic, or Business Operations Center, do this:

   a. Ensure BRM images are available and BRM is deployed successfully, for PDC pods to deploy successfully

   b. Direct WebLogic Kubernetes Operator to monitor the BRM namespace:

```
helm upgrade weblogic-operator  weblogic-operator/weblogic-operator \
--namespace Operator \
--reuse-values \
--set "domainNamespaces={BrmNameSpace}" \
--wait
```

where *Operator* is the namespace you created for WebLogic Kubernetes Operator as part of the prerequisite tasks.

c. Create WebLogic domains by entering this command from the **helmcharts** directory:

```
helm install OpJobReleaseName oc-cn-op-job-helm-chart --namespace BrmNameSpace --
values OverrideValuesFile
```

where *OpJobReleaseName* is the release name for **oc-cn-op-job-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.

4. Install BRM cloud native services by entering this command from the **helmcharts** directory:

```
helm install BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
BrmNameSpace
```

where *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance. It must be different from the one used for **oc-cn-init-db-helm-chart**.

5. To install the ECE cloud native service, enter this command from the **helmcharts** directory:

```
helm install EceReleaseName oc-cn-ece-helm-chart --namespace BrmNameSpace --values
OverrideValuesFile
```

where *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.

12

Deploying into Oracle Cloud Infrastructure

Learn how to deploy Oracle Communications Billing and Revenue Management (BRM) 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 computing capabilities (as physical hardware instances) and storage capacity in a flexible overlay virtual network that is securely accessible from your on-premise network. BRM cloud native deployment is tested in Oracle Cloud Infrastructure for the following services both on Virtual Machine and Bare Metal:

- BRM cloud native application and database running on IaaS
- BRM cloud native application managed by Oracle Kubernetes Engine and database on IaaS
- BRM cloud native application managed by Oracle Kubernetes Engine and database on DBaaS

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

Note:

These are the bare minimum tasks for deploying BRM 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 database system on a bare metal or virtual machine instance.

   Select a database version that is compatible with the BRM cloud native software requirements. See *BRM Compatibility Matrix*.
3. 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 BRM cloud native software requirements.
4. 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.
5. Install and configure **kubectl** on your system to perform operations on your cluster in Oracle Cloud Infrastructure.

6. 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 --token-version 2.0.0
   ```

   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.

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

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

8. 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
   ```

9. Download and configure Helm in your local system.

10. Place the BRM cloud native Helm chart on your system where you have downloaded and configured **kubectl** and Helm. Then, follow the instructions in "Configuring and Deploying BRM Cloud Native" in *BRM Cloud Native Deployment Guide*.

13

Uninstalling Your BRM Cloud Native Deployment

Learn how to uninstall the Oracle Communications Billing and Revenue Management (BRM) cloud native deployment from your system.

Topics in this document:

Uninstalling Your BRM Cloud Native Deployment

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

To uninstall, enter this command:

```
helm delete ReleaseName --namespace NameSpace
```

where:

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

Uninstalling Selected BRM Cloud Native Services

Depending on the scenario, you might need to temporarily or permanently uninstall BRM cloud native services, such as Billing Care, the Billing Care REST API, or Business Operations Center, while retaining other BRM services. To do this, you upgrade your **oc-cn-helm-chart** release by disabling the service you intend to remove.

For example, to remove only the Billing Care REST API service, you would set the **ocbc.bcws.isEnabled** key to **false** in your **override-values.yaml** file and then upgrade your release of **oc-cn-helm-chart**:

```
helm upgrade --namespace NameSpace ReleaseName oc-cn-helm-chart --values override-
values.yaml
```

This would bring down the WebLogic servers that are hosting the Billing Care REST API and remove all resources created for this service through **oc-cn-helm-chart**.

Part IV
Customizing BRM Cloud Native

This part provides information about customizing Oracle Communications Billing and Revenue Management (BRM) cloud native. It contains the following chapters:

14

Customizing BRM Cloud Native Services

Learn how to customize the Oracle Communications Billing and Revenue Management (BRM) server and clients in a cloud native environment to meet your business needs.

The Podman build commands in this chapter reference Dockerfile and related scripts as is from the **oc-cn-docker-files-15.1.*x.x.x*.tgz** package. Ensure you use your own version of Dockerfile and related scripts before running the build command.

Topics in this document:

- Customizing BRM Server
- Customizing Billing Care
- Customizing ECE

Caution:

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

Customizing BRM Server

You can customize BRM Server by layering the BRM cloud native image with a customized library file.

For example, you could extend the **fm_subsciption_pol_custom.so** library file and layer it with the BRM cloud native image by doing this:

1. Customize your **lib/fm_subscription_pol_custom.so** library file as follows:
   a. Enable the BRM SDK by setting the following keys in your **override-values.yaml** file for **oc-cn-helm-chart**:

      ```
      brm_sdk:
         isEnabled: true
         deployment:
            imageName: brm_sdk
            imageTag: 15.1.x.x.x
         volume:
            storage: 50Mi
      ```

   b. By default, the brm-sdk pod uses dynamic volume provisioning. To use a static volume instead, add the **createOption** keys under **brm_sdk.volume**. For example:

      ```
      brm_sdk:
        volume:
          storage: 50Mi
          createOption:
      ```

```
      hostpath:
        path: pathOnNode
        type: Directory
```

where *pathOnNode* is the location on the host system of the external PV.

c. Run the **helm upgrade** command to deploy the brm-sdk pod:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values
OverrideValuesFile --namespace BrmNameSpace
```

d. Run the following **kubectl** command to retrieve the brm-sdk pod name:

```
kubectl get pods --namespace BrmNameSpace | grep brm-sdk
```

If successful, you should see something similar to this:

```
NAME                           READY    STATUS     RESTARTS    AGE
brm-sdk-f67b95777-bf8j5        1/1      Running    0           18m
```

e. Run the following **kubectl** command to retrieve the name of the PVC volume for brm-sdk:

```
kubectl get pvc --namespace BrmNameSpace | grep brm-sdk
```

If successful, you should see something similar to this:

```
NAME       STATUS    VOLUME                                     CAPACITY
ACCESS MODES    STORAGECLASS    AGE
brm-sdk    Bound     pvc-094feae0-4d11-4887-83a0-b47a0fc6a3f4   50Mi
RWX             myclass         23h
```

f. List the files and folders in **/mnt/oke_test/brm** to verify that the PVC volume is present:

```
ls /mnt/oke_test/brm/ | grep pvc-094feae0-4d11-4887-83a0-b47a0fc6a3f4
```

If successful, you should see something similar to this:

```
brm-sdk-pvc-094feae0-4d11-4887-83a0-b47a0fc6a3f4
```

g. Do one of the following:

- Copy the custom C file to the PVC:

  ```
  cp customFile nfsMountPath/BrmNameSpace/pvcVolumePath/
  ```

  For example:

  ```
  cp fm_cust_pol_valid_billinfo.c /mnt/oke_test/brm/brm-sdk-
  pvc-094feae0-4d11-4887-83a0-b47a0fc6a3f4/
  ```

- Copy the custom C file to the **oc-cn-helm-chart/brm_sdk_scripts/** directory and run the **helm upgrade** command:

  ```
  cp customFile  oc-cn-helm-chart/brm_sdk_scripts/
  helm upgrade BrmReleaseName oc-cn-helm-chart --namespace
  BrmNameSpace --values oc-cn-helm-chart/override_values.yaml
  ```

  For example:

  ```
  cp fm_cust_pol_valid_billinfo.c  oc-cn-helm-chart/brm_sdk_scripts/
  helm upgrade release oc-cn-helm-chart --namespace brm --values oc-
  cn-helm-chart/override_values.yaml
  ```

  The files from **oc-cn-helm-chart/brm_sdk_scripts/** will be present at **/oms/load** in the brm-sdk pod.

h. Run the **kubectl** command to get a shell to a running container:

   ```
   kubectl exec --namespace BrmNameSpace -it brmSDKPod bash
   ```

   For example:

   ```
   kubectl exec --namespace brm -it brm-sdk-f67b95777-bf8j5 bash
   ```

i. Build your custom library file in one of these ways:

   - If you copied your custom C file to the PVC in step 1.g, do this:

     ```
     cd source/sys/libraryName
     cp /oms/ext/fileName .
     make
     ```

     For example:

     ```
     cd source/sys/fm_cust_pol/
     cp /oms/ext/fm_cust_pol_valid_billinfo.c .
     make
     ```

   - If you copied your custom C file to **oc-cn-helm-chart/brm_sdk_scripts/** in step 1.g, do this:

     ```
     cd source/sys/libraryName
     cp /oms/load/fileName .
     make
     ```

     For example:

     ```
     cd source/sys/fm_cust_pol/
     cp /oms/load/fm_cust_pol_valid_billinfo.c .
     make
     ```

j. Copy the generated library file to the PVC:

   ```
   cp customLibrary.so /oms/ext/
   ```

   For example:

   ```
   cp fm_cust_pol_custom.so /oms/ext/
   ```

k. Copy the library file from the PVC to the **$PIN_HOME/lib** directory:

   ```
   cp nfsMountPath/BrmNameSpace/brmSDKPod/customLibrary.so $PIN_HOME/lib
   ```

   For example:

   ```
   cp /mnt/oke_test/brm/brm-sdk-pvc-094feae0-4d11-4887-83a0-b47a0fc6a3f4/
   fm_cust_pol_custom.so $PIN_HOME/lib
   ```

2. Build the custom Connection Manager (CM) image using the Dockerfile:

   ```
   FROM cm:15.1.x.x.x
   USER root
   COPY lib/fm_subscription_pol_custom.so $PIN_HOME/lib/fm_subscription_pol_custom.so
   RUN chown -R omsuser:root $PIN_HOME/lib/fm_subscription_pol_custom.so && \
    chmod 755 ${PIN_HOME}/lib/fm_subscription_pol_custom.so
   USER omsuser
   ```

3. Build the BRM Server image by entering this command:

   ```
   podman build --format docker --tag cm:imageTag --file Dockerfile_cm .
   ```

4. Push the image to the image repository:

   ```
   podman tag cm:imageTag imageRepository/cm:imageTag
   podman push imageRepository/cm:imageTag
   ```

5. Update the custom image name in the **override-values.yaml** file. For example:

   ```
   cm:
      isEnabled: true
      deployment:
         replicaCount: 1
         imageName: cm
         imageTag: imageTag
   ```

   where *imageTag* must match the value used in step 4.

6. Add the custom configuration for the CM **pin.conf** to the **configmap_pin_conf_cm.yaml** file.

   For example, for the **fm_cust_pol_custom** library:

   ```
   - cm fm_module ${PIN_HOME}/lib/fm_cust_pol_custom${LIBRARYEXTENSION}
   fm_cust_pol_custom_config fm_cust_pol_init pin
   ```

7. Run the **helm upgrade** command to update the release with the new CM image:

```
helm upgrade BrmReleaseName oc-cn-helm-chart --namespace BrmNameSpace --
values oc-cn-helm-chart/override_values.yaml
```

Customizing Billing Care

Extensibility is one of the biggest features of on-premise Billing Care, and this same extensibility is available in the Billing Care cloud native deployment. You can override the existing Billing Care behavior, such as changing labels and icons, add new flows and screens, and so on, by using the Billing Care SDK.

To use the Billing Care SDK in a cloud native environment, do this:

1. Build the Billing Care SDK WAR the same way as described in "Packaging and Deploying Customizations" in *Billing Care SDK Guide*.
2. Create a Billing Care SDK image by using the Linux image as a base.
3. Update the **override-values.yaml** file to direct **oc-cn-op-job-helm-chart** to deploy the SDK WAR file and link it with Billing Care or the Billing Care REST API WAR after deploying them.

The cloud native package includes all of the scripts necessary to prepare and run an SDK image. For example, if your SDK WAR is named **BillingCareCustomizations.war**, you would build the Billing Care SDK image like this:

1. Go to the **oc-cn-docker-files/ocbc/billing_care_sdk** directory.
2. Copy the **BillingCareCustomizations.war** file to your current working directory (**oc-cn-docker-files/ocbc/billing_care_sdk**).
3. Build the SDK image by entering this command:

```
podman build --format docker --build-arg SDK_WAR=BillingCareCustomizations.war --tag
oracle/billingcare_sdk:15.1.x.x.x  .
```

4. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, edit the keys shown in Table 14-1. This directs **oc-cn-op-job-helm-chart** to deploy the Billing Care SDK image rather than the Billing Care image and to include additional files that are needed for successful deployment of SDK.

Table 14-1 Billing Care SDK Keys

Key	Path	Description
imageName	**ocbc.bc.deployment.sdk** **ocbc.bcws.deployment.sdk**	Set this to **oracle/billingcare_sdk**. This is the name of the image, which must be used for the billingcare pod.
imageTag	**ocbc.bc.deployment.sdk** **ocbc.bcws.deployment.sdk**	Set this to **15.1.0.0.0**. This tags the image used for the billingcare pod.
isEnabled	**ocbc.bc.sdk** **ocbc.bcws.sdk**	Set this key to **true** if you want to deploy SDK.

Table 14-1 (Cont.) Billing Care SDK Keys

Key	Path	Description
deployName	**ocbc.bc.sdk** **ocbc.bcws.sdk**	The name of the SDK Library in the **Manifest.MF** file. The default is **BillingCareCustomizations**.

5. Install **oc-cn-op-job-helm-chart** followed by **oc-cn-helm-chart** to customize Billing Care or the Billing Care REST API with SDK.

Customizing ECE

You can customize the ECE image by layering the native image with the customized code.

For example:

```
 > cat Dockerfile_custom_ece
FROM oc-cn-ece:15.1.x.x.x
USER root
#commands that need to be run
USER eceuser
```

To build the image, run this Podman command:

```
podman build --format docker --tag customECE:15.1.x.x.x --file customECEDockerfile .
```

where *customECE* is the name of your custom ECE Helm chart, and *customECEDockerfile* is the name of your custom Dockerfile.

For the Helm chart to take the new custom image for installation, set these keys in your **override-values.yaml** file for the ECE Helm chart:

```
imageRepository: "imageRepo:imagePort"
container:
   image: "customECEImageName"
```

15
Building Your Own Images

Learn how to build your own images of the Oracle Communications Billing and Revenue Management (BRM), Elastic Charging Engine (ECE), Pipeline Configuration Center, Pricing Design Center (PDC), Billing Care, and Business Operations Center applications.

The Podman build commands in this chapter reference Dockerfile and related scripts as is from the **oc-cn-docker-files-15.1.***x.x.x***.tgz** package. Ensure you use your own version of Dockerfile and related scripts before running the build command.

Topics in this document:

Sample Dockerfiles included in the BRM cloud native deployment package (**oc-cn-docker-files-15.1.***x.x.x***.tgz**) are examples that depict how default images are built for BRM. 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 images. Support is restricted to core product issues only and no support will be provided for custom Dockerfiles and scripts.

Building BRM Server Images

To build images for BRM Server, your staging area ($PIN_HOME) must be available from where the images are built. After you unpack **oc-cn-docker-files-15.1.***x.x.x***.tgz**, the BRM Server directory structure will be **oc-cn-docker-files/ocbrm**.

Note:

If you are using Podman to build your images, pass the **--format docker** flag with the **podman build** command.

Building your own BRM Server images involves these high-level steps:

1. You build the BRM Server base image. See "Building Your BRM Server Base Image".
2. You build images for each BRM Server component. See "Building Images of BRM Server Components".
3. You build the Web Services Manager image. See "Building Web Services Manager Images".
4. You build the BRM REST Services Manager image. See "Building BRM REST Services Manager Images".
5. You containerize the Email Data Manager. See "Containerization of Email Data Manager".
6. You containerize the roaming pipeline. See "Containerization of Roaming Pipeline".
7. You build and deploy Vertex Manager. See "Building and Deploying Vertex Manager".

Building Your BRM Server Base Image

To make your directory structure ready for building base images:

1. Edit the **$PIN_HOME/bin/orapki** binary to replace the staging Java path with ${JAVA_HOME}.
2. Create the **$PIN_HOME/installer** directory.
3. If you're behind a proxy server, set the **$PROXY** variable:

   ```
   export PROXY=ProxyHost:Port
   ```

4. Download the Java binary and then copy it to **$PIN_HOME**. See "BRM Software Compatibility" for the latest supported version of Java.
5. Download the Perl binary and then copy it to **$PIN_HOME**. See "BRM Software Compatibility" for the latest supported version of Perl.
6. For your database client:
   a. Copy **oracle_client_response_file.rsp** (64 bit), **downloadOracleClient.sh**, and **waitForOracleClientInst.sh** from **oc-cn-docker-files/ocbrm/base_images** to **$PIN_HOME**.
   b. Modify these parameters in the **downloadOracleClient.sh** file:
      - ORACLE_CLIENT_ZIP: Enter the binary name.
      - REPOSITORY_URL: Enter the location to fetch the database client binary.
   c. If the db_client binary is already downloaded, copy the binary to the **$PIN_HOME/installer** directory.

After preparing your directory structure, build your BRM Server base image:

- For database client 12CR2 (64 Bit) + Java + Perl, enter this command:

  ```
  podman build --format docker --build-arg PROXY=$PROXY --tag
  db_client_and_java_perl:15.1.x.x.x --file DockerFileLocation/
  Dockerfile_db_client_and_java_perl .
  ```

- For database client 12CR2 (64 Bit) + Java, enter this command:

  ```
  podman build --format docker --build-arg PROXY=$PROXY --tag
  db_client_and_java:15.1.x.x.x --file DockerFileLocation/
  Dockerfile_db_client_and_java .
  ```

- For Java, enter this command:

```
podman build --format docker --build-arg PROXY=$PROXY --tag java:15.1.x.x.x --file
DockerFileLocation/Dockerfile_java .
```

- For Java + Perl, enter this command:

```
podman build --format docker --build-arg PROXY=$PROXY --tag java_perl:15.1.x.x.x --
file DockerFileLocation/Dockerfile_java_perl .
```

Note:

If the existing database is used with custom build images, do this:

- Override the **ocbrm.use_oracle_brm_images** key in the Helm chart with a value of **false**.
- Set the **ocbrm.existing_rootkey_wallet** key to **true**.
- Copy your client wallet files to the **oc-cn-helm-chart/existing_wallet** directory.

Building Images of BRM Server Components

The **oc-cn-docker-files-15.1.*x.x.x*.tgz** package includes references to all of the Dockerfiles and scripts needed to build images of BRM Server components (except for **oraclelinux:8**).

To build an image of a BRM Server component:

1. Copy these scripts from the **oc-cn-docker-files/ocbrm** directory to your staging area at **$PIN_HOME**:
   - **entrypoint.sh**
   - **createWallet.sh**
   - **cm/preStopHook.sh_cm**
   - **cm/postStartHook.sh**
   - **cm/updatePassword.sh**
   - **eai_js/preStopHook.sh_eai**
2. Do one of these:
   - For the batch pipeline, roaming pipeline, and real-time pipeline, copy **entrypoint.sh** and **createWallet.sh** to **$PIN_HOME/..**, and copy **$PIN_HOME/../setup/BRMActions.jar** to the **$PIN_HOME/jars** directory for building the images.
   - For all other components, copy the **$PIN_HOME/../setup/BRMActions.jar** file to **$PIN_HOME**.
3. Set these environment variables:
   - **$PIN_HOME:** Set this to your staging area.
   - **$PERL_HOME:** Set this to the path of Perl. See "BRM Software Compatibility" for the latest supported version of Perl.
   - **$JAVA_HOME:** Set this to the Java path. See "BRM Software Compatibility" for the latest supported version of Java.
4. Build the image for your BRM component.

   For example, to build a CM image, you'd enter this:

```
podman build --format docker --tag cm:15.1.x.x.x --build-arg
STAGE_PIN_HOME=$PIN_HOME --build-arg STAGE_JAVA_HOME=$JAVA_HOME --build-arg
STAGE_PERL_HOME=$PERL_HOME --file DockerfileLocation/Dockerfile .
```

To build a roaming pipeline image, you'd enter this:

```
podman build --format docker --tag roam_pipeline:$BRM_VERSION --build-arg
STAGE_PERL_HOME=StagePerlPath .
```

where *StagePerlPath* is the path to the Perl files in your staging area at $PIN_HOME.

To build a dm-oracle image, you'd enter this:

```
podman build --format docker --force-rm=true --no-cache=true --tag
dm_oracle:15.1.x.x.x --file DockerfileLocation/Dockerfile .
```

where *DockerfileLocation* is the path to the Dockerfiles for your BRM component.

Note:

Build batch and realtime pipeline images from the **$PIN_HOME/..** directories.

Building Web Services Manager Images

To containerize images for Web Services Manager, your staging area ($PIN_HOME) must be available from where the Docker images are built.

You can create one of these Web Services Manager containers:

- Building and Deploying Web Services Manager for Helidon Image
- Building and Deploying Web Services Manager for Apache Tomcat Image
- Building and Deploying Web Services Manager for WebLogic Server Image

Building and Deploying Web Services Manager for Helidon Image

To build and deploy the Web Services Manager for Helidon image:

1. Copy these files from the **oc-cn-docker-files** directory to your staging area at **$PIN_HOME**:
   - **brm_wsm_entrypoint.sh**
   - **Dockerfile**
   - **mkstore** (**oc-cn-docker-files/base_images**)
2. Build the Web Services Manager image by entering this command:

   ```
   podman build --format docker --tag brm_wsm:$BRM_VERSION .
   ```
3. Configure your Web services by updating the **configmap_brm_wsm_props.yaml** file.
4. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **ocbrm.wsm.soap.isEnabled** key to **true**.
5. Deploy the BRM Helm chart:

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

   where:

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

Building and Deploying Web Services Manager for Apache Tomcat Image

The Web Services Manager Dockerfile is based on the official Apache Tomcat image. The sample Web Services Manager Dockerfile includes both the XML element-based and XML string-based SOAP Web Services implementation. Use this Dockerfile to build an image that can call any standard BRM opcode that is exposed as a SOAP Web service.

The Web Services Manager **Infranet.properties** configuration is available as a Kubernetes ConfigMap. To expose a custom opcode as a Web service, place your customized WAR filepath in the Dockerfile. When multiple pod replicas are configured, each pod runs its own copy of Apache Tomcat. By default, Web Services Manager is exposed as a Kubernetes NodePort service running on port 30080.

Containerizing the Web Services Manager for Tomcat image involves these high-level steps:

1. Building the Web Services Manager Tomcat Image
2. Deploying the Web Services Manager Tomcat Image in Kubernetes

Building the Web Services Manager Tomcat Image

To build the Web Services Manager for Apache Tomcat image:

1. Download the JAX-WS reference implementation JARs from *JAX-WS Java API for XML Web Services* (https://javaee.github.io/metro-jax-ws/).
2. Copy the **jaxws-ri-2.3.*x*.zip** file to your staging area at $PIN_HOME.
3. Unzip the **jaxws-ri-2.3.*x*.zip** file.
4. Download Apache Tomcat 9 from the Apache Tomcat website:
   https://tomcat.apache.org/download-90.cgi

   See *BRM Compatibility Matrix* for information about compatible versions of Apache Tomcat.
5. Copy **apache-tomcat-9.*x*.tar.gz** to your staging area at $PIN_HOME.
6. Copy these files from the **oc-cn-docker-files** directory to your staging area at $PIN_HOME.
   - **wsm_entrypoint.sh**
   - **Dockerfile**
   - **context.xml**
   - **BRMActions.jar**
7. Update Tomcat in the Dockerfile to the latest version.
8. Build the Web Services Manager image by entering this command:

   ```
   podman build --format docker --tag brm_wsm:$BRM_VERSION .
   ```

Deploying the Web Services Manager Tomcat Image in Kubernetes

To deploy the Web Services Manager for Tomcat image in Kubernetes:

1. Configure your Web services by updating the **configmap_infranet_properties_wsm.yaml** file.
2. In the **override-values.yaml** file for **oc-cn-helm-chart**, set the following values:
   - **ocbrm.wsm.deployment.tomcat.isEnabled**: Set this to **true**.
   - **ocbrm.wsm.deployment.tomcat.walletPassword**: Set this to the Base64-encoded wallet password for the Web Services Manager image.
   - **ocbrm.wsm.deployment.tomcat.basicAuth**: Optionally, set this to **true** to enable BASIC authentication.
3. Optionally, for BASIC authentication, configure users in the **wsm_config/tomcat-users.xml** file for **oc-cn-helm-chart**:
   a. Open **tomcat-users.xml** in a text editor.
   b. Locate the following lines and specify the login details of the user:

      ```
      <role rolename="role"/>
      <user username="username" password="password" roles="role"/>
      ```

      where:
      - *role* is the role with permissions to access Web services, for example, **brmws**.
      - *username* is the user name for accessing Web services.
      - *password* is the password for accessing Web services.
   c. Save and close the file.

   See "User File Format" under *MemoryRealm* in the Apache Tomcat documentation for more information about the format of **tomcat-users.xml**.
4. Deploy the BRM Helm chart:

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

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

Building and Deploying Web Services Manager for WebLogic Server Image

To deploy and use Web Services Manager on WebLogic Server, you should be familiar with:

- Oracle WebLogic Server 12.2.1.3. See the Oracle WebLogic Server 12.2.1.3 documentation (https://docs.oracle.com/middleware/12213/wls/index.html).
- Oracle WebLogic Kubernetes Operator. See the WebLogic Kubernetes Operator documentation (https://oracle.github.io/weblogic-kubernetes-operator/).

The image for deploying BRM Web Services Manager on Oracle Weblogic Server 12.2.1.3 uses the domain in image approach. The image includes a WebLogic domain named **brmdomain**. When you build the image, the BRM SOAP Web Services application WAR files get deployed in this domain.

Containerizing the Web Services Manager for WebLogic Server image involves these high-level steps:

1. Building the Web Services Manager WebLogic Image
2. Deploying the Web Services Manager WebLogic Image in Kubernetes
3. Updating the BRM Web Services Manager Configuration
4. Restarting the WebLogic Server Pods
5. Scaling Your WebLogic Managed Server

Building the Web Services Manager WebLogic Image

The BRM Web Services Manager on WebLogic Server image uses two images that run two containers inside each WebLogic Server pod.

To build the **brm_wsm_wls15.1.***x.x.x* image:

1. Copy the contents of the **oc-cn-docker-files/ocbrm/brm_soap_wsm/weblogic/dockerfiles** directory to your staging area at $PIN_HOME.
2. Customize the WebLogic domain-related properties by editing the **dockerfiles/properties/docker-build/domain.properties** file. For example:

   ```
   DOMAIN_NAME=brmdomain
   ADMIN_PORT=7111
   ADMIN_NAME=admin-server
   ADMIN_HOST=wlsadmin
   MANAGED_SERVER_PORT=8111
   MANAGED_SERVER_NAME_BASE=managed-server
   CONFIGURED_MANAGED_SERVER_COUNT=3
   CLUSTER_NAME=cluster-1
   DEBUG_PORT=8453
   DB_PORT=1527
   DEBUG_FLAG=true
   PRODUCTION_MODE_ENABLED=true
   CLUSTER_TYPE=DYNAMIC
   T3_CHANNEL_PORT=30012
   T3_PUBLIC_ADDRESS=kubernetes
   IMAGE_TAG=brm_wsm_wls:$BRM_VERSION
   ```

3. Set the WebLogic domain user name and password by editing the **dockerfiles/properties/docker-build/domain_security.properties** file. For example:

   ```
   username=UserName
   password=Password
   ```

   Note:

   It is strongly recommended that you set a new user name and password when building the image.

   For details about securing the **domain_security.properties** file, see https://github.com/oracle/docker-images/tree/master/OracleWebLogic/samples/12213-domain-home-in-image.

4. Build the **brm_wsm_wls:15.1.***x.x.x* image by running the **build.sh** script.

   The script creates an image based on the custom tag defined in **dockerfiles/properties/docker-build/domain.properties**. By default, it creates the **brm_wsm_wls:15.1.***x.x.x* image and then deploys the **BRMWebServices.war** and **infarnetwebsvc.war** files.

> **Note:**
>
> If you don't want to deploy either **BRMWebServices.war** or **infarnetwebsvc.war**, modify the **dockerfiles/container-scripts/app-deploy.py** script.

5. Build the **brm_wsm_wl_init:15.1.***x.x.x* image by running this command:

   ```
   podman build --format docker --tag brm_wsm_wl_init:15.1.x.x.x --file
   Dockerfile_init_wsm .
   ```

   This image runs an init container, which populates the Oracle wallet that is used by Web Services Manager to connect to the CM.

Deploying the Web Services Manager WebLogic Image in Kubernetes

You deploy the WebLogic Operator Helm chart so that Web Services Manager can work in a Kubernetes environment.
To deploy the Web Services Manager for WebLogic Server image in Kubernetes:

1. Clone the Oracle WebLogic Kubernetes Operator Git project:

   ```
   git clone https://github.com/oracle/weblogic-kubernetes-operator
   ```

2. Modify these keys in the **override-values.yaml** file for **oc-cn-helm-chart**:

   > **Note:**
   >
   > Ensure that you set the **wsm.deployment.weblogic.isEnabled** key to **true**.

   ```
   wsm:
      deployment:
         weblogic:
            isEnabled: true
            imageName: brm_wsm_wls
            initImageName: brm_wsm_wl_init
            imageTag: $BRM_VERSION
            username: user_name
            password: password
            replicaCount: 1
            adminServerNodePort:
            log_enabled: false
            minPoolSize: 1
            maxPoolSize: 8
            poolTimeout: 30000
   ```

3. If the WebLogic user name and password was updated when building the **brm_wsm_wls:15.1.***x.x.x* image, also update the base64-encoded WebLogic user name and password in these keys:

   ```
   .Values.ocbrm.wsm.deployment.weblogic.username
   .Values.ocbrm.wsm.deployment.weblogic.password
   ```

4. Add the BRM WebLogic Server namespace in the **kubernetes/charts/weblogic-operator/values.yaml** file:

   ```
   domainNamespaces:
     - "default"
     - "NameSpace"
   ```

5. Deploy the WebLogic Operator Helm chart:

   ```
   helm install weblogic-operator kubernetes/charts/weblogic-operator --namespace
   WebOperatorNameSpace --values WebOperatorOverrideValuesFile --wait
   ```

   where:

   - *WebOperatorNameSpace* is the namespace in which to create WebLogic Operator Kubernetes objects.
   - *WebOperatorOverrideValuesFile* is the path to a YAML file that overrides the default configurations in the WebLogic Operator Helm chart's **values.yaml** file.

6. Deploy the BRM helm chart:

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

   where:

   - *ReleaseName* is the release name, which is used to track this installation instance.
   - *NameSpace* is the namespace in which **oc-cn-helm-chart** will be installed.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the BRM Helm chart's **values.yaml** file.

Updating the BRM Web Services Manager Configuration

Update the basic configurations for BRM Web Services Manager by editing the Kubernetes ConfigMap (**configmap_infranet_properties_wsm_wl.yaml**). After updating the configuration, restart your WebLogic Server pods.

Restarting the WebLogic Server Pods

To restart your WebLogic Server pods:

1. Stop the WebLogic Server pods by doing this:
   a. In the **domain_brm_wsm.yaml** file, set the **serverStartPolicy** key to **NEVER**.
   b. Update your Helm release.

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

      where *NameSpace* is the namespace in which **oc-cn-helm-chart** will be installed.
2. Start the WebLogic Server pods by doing this:
   a. In the **domain_brm_wsm.yaml** file, set the **serverStartPolicy** key to **IF_NEEDED**.
   b. Update your Helm release:

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

Scaling Your WebLogic Managed Server

The default configuration starts one WebLogic Managed Server pod. To modify the configuration to start up to three pods, do this:

1. In the **oc-cn-helm-chart/values.yaml** file, set the **.Values.ocbrm.wsm.deployment.weblogic.replicaCount** key to **1**, **2**, or **3** WebLogic Managed Server pods.

2. Update your Helm release:

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

You set the maximum number of managed servers in the BRM Web Services Manager image by modifying the **CONFIGURED_MANAGED_SERVER_COUNT** property in the **dockerfiles/properties/docker-build/domain.properties** file.

Containerization of Email Data Manager

The Email Data Manager (DM) enables you to send customer notifications and invoices to your customers through email automatically. The Email DM uses the Sendmail client to forward emails to Postfix, which is the SMTP server. In-turn, Postfix sends the emails to your customers.

The Email DM will have the Sendmail client, and the Kubernetes host will have Postfix running. You must install and configure Postfix on your Kubernetes host.

To configure your cm pod to point to the Email DM, add this key to the **oc-cn-helm-chart/values.yaml** file:

```
ocbrm.dm_email.deployment.smtp: EmailHostName
```

where *EmailHostName* is the hostname of the server on which the Email DM is deployed. For example: em389.us.example.com.

To configure the Kubernetes host or SMTP server to accept data from the Email DM, do this:

1. Log in as the root user to the Kubernetes host.
2. Add the IP address for the Kubernetes host to the **/etc/postfix/main.cf** file:

   ```
   inet_interfaces=localhost, HostIPAddress
   ```

   For example, if the Kubernetes host is 10.242.155.149.

   ```
   inet_interfaces=localhost, 10.242.155.149
   ```

3. Retrieve the container network configuration by running this command on the Kubernetes host:

   ```
   /sbin/ifconfig cni0 | grep netmask | awk '{print$2"\n"$4}'
   ```

   The output will be similar to this:

   ```
   10.244.0.1 ← The Kubernetes host IP, which is in the container network.
   255.255.255.0
   ```

4. Edit the **mynetworks** field in the **/etc/postfix/main.cf** file to include the Kubernetes network in the list of trusted SMTP clients. If the Kubernetes host IP and Email DM container IP are in different networks, add both networks to the **mynetworks** field:

   ```
   mynetworks = TrustedNetworks
   ```

   where *TrustedNetworks* is the IP addresses for the SMTP clients that are allowed to relay mail through Postfix.

   For example:

   ```
   mynetworks = 168.100.189.0/28, 127.0.0.0/8, 10.244.0.0/24
   ```

5. Do one of these:
   - If Postfix is already running in the host, run this command:

```
systemctl restart postfix.service
```

- If Postfix isn't running in the host, run this command:

  ```
  systemctl start postfix.service
  ```

> **Note:**
>
> In case of a multi-node environment, you can configure Postfix on the primary node (or any one node).

Containerization of Roaming Pipeline

Roaming allows a wireless network operator to provide services to mobile customers from another wireless network. For example, when a mobile customer makes a phone call from outside the home network, roaming allows the customer to access the same wireless services that he has with his home network provider through a visited wireless network operator.

You feed the input files for the roaming pipeline through a Kubernetes PersistentVolumeClaim (PVC). The EDR output files will be available in a PVC for consumption of the rel-daemon pod. When building the roaming pipeline image, pass the Perl path in these files as part of **build-arg**.

To containerize the roaming pipeline, update the **configmap_infranet_properties_rel_daemon.yaml** file to specify how to load your rated CDR output files. For example:

```
batch.random.events TEL, ROAM
ROAM.max.at.highload.time 4
ROAM.max.at.lowload.time  2
ROAM.file.location /oms/ifw/data/roamout
ROAM.file.pattern test*.out
ROAM.file.type STANDARD
```

> **Note:**
>
> The input file to the splitter pipeline must start with **Roam_**.

Building and Deploying Vertex Manager

To deploy Vertex Manager (dm-vertex), you layer the dm-vertex image with the libraries for Vertex Communications Tax Q Series (CTQ) or Vertex Sales Tax Q Series (STQ). For the list of supported library versions, see *BRM Compatibility Matrix*.

Deploying with Vertex Communications Tax Q Series

You deploy Vertex Manager with Vertex CTQ by doing the following:

1. Building the new Vertex Manager image by layering it with Vertex CTQ libraries.
   a. Copy the entire Vertex CTQ installation directory to the **$PIN_HOME** directory, where **$PIN_HOME** is set to the path of your staging area.

b. Update the paths in the **64bit/bin/ctqcfg.xml**, **64bit/cfg/ctqcfg.xml**, and other Vertex CTQ files present in the Vertex CTQ installation directory. For example:

```
<configuration name="CTQ Test">
<fileControl>
  <updatePath>/oms/vertex/64bit/dat</updatePath>
  <archivePath>/oms/vertex/64bit/dat</archivePath>
  <callFilePath>/oms/vertex/64bit/dat</callFilePath>
  <reportPath>/oms/vertex/64bit/rpt</reportPath>
  <logPath>/oms/vertex/64bit/log</logPath>
</fileControl>
```

c. In your copied Vertex CTQ installation directory, update the **64bit/bin/odbc/odbc.ini** file. For example:

> **Note:**
>
> Set the **Driver** and **TNSNamesFile** entries to the file system path inside the pod.

```
[CtqTestOracle]
Description=Vertex, Inc. 8.0 Oracle Wire Protocol
Driver=/oms/vertex/64bit/bin/odbc/lib/VXor827.so
…
HostName=DBhostname
LogonID=DBuser
PortNumber=1521
Password=DBpassword
ServerName=//IPaddress:1521/DBalias
SID=DBalias
TNSNamesFile=/oms/ora_k8/tnsnames.ora
```

where:

- *DBhostname* is the host name of the machine on which the Vertex tax calculation database is installed.
- *DBuser* is the Vertex database schema user name.
- *DBpassword* is the password for the Vertex database schema user.
- *IPaddress* is the IP address of the machine on which the Vertex tax calculation database is installed.
- *DBalias* is the Vertex database alias name, which is defined in your **tnsnames.ora** file.

d. Layer the default images provided by Oracle.

For example, to layer dm-vertex with Vertex CTQ, you could add these sample commands to its Dockerfile. In this example, **$PIN_HOME** is set to **/oms** inside the pod.

```
FROM dm_vertex:15.1.x.x.x

USER root
RUN mkdir -p /oms/vertex/64bit/cfg
RUN chown -R omsuser:root /oms/vertex/64bit/cfg
COPY ./Vertex_CTQ_30206/ /oms/vertex
COPY Vertex_CTQ_30206/64bit/lib/libctq.so /oms/lib/
COPY Vertex_CTQ_30206/64bit/bin/odbc/lib/libodbc.so /oms/lib/libodbc.so
```

```
RUN chown -R omsuser:root /oms/vertex
RUN chown -R omsuser:root /oms/lib/libctq.so
RUN chown -R omsuser:root /oms/lib/libodbc.so
USER omsuser
```

e. Build your new Vertex Manager image. For example:

```
podman build --format docker --tag dm_vertex_ctq:15.1.x.x.x --file
Dockerfile_vertex_ctq .
```

2. Enabling and configuring Vertex Manager in your BRM cloud native deployment.

   a. Set these environment variables in your **oc-cn-helm-chart/templates/dm_vertex.yaml** file:

```
- name: LD_LIBRARY_PATH
  value: "/oms/vertex/64bit/bin/odbc:/oms/lib:/oms/sys/dm_vertex:/oms/vertex/
64bit/lib"
- name: CTQ_CFG_HOME
  value: "/oms/vertex/64bit/bin"
- name: ODBCINI
  value: "/oms/vertex/64bit/bin/odbc/odbc.ini"
```

   b. Uncomment these entries in your **oc-cn-helm-chart/templates/configmap_pin_conf_dm_vertex.yaml** file:

```
- dm_vertex commtax_sm_obj ${DM_VERTEX_CTQ_SM}
- dm_vertex commtax_config_name ${DM_VERTEX_CTQ_CFG_NAME}
- dm_vertex commtax_config_path ${DM_VERTEX_CTQ_CFG_PATH}
```

   c. Update these key in your **override-values.yaml** file for **oc-cn-helm-chart**:

```
dm_vertex:
  isEnabled: true
  deployment:
    replicaCount: 1
    imageName: dm_vertex_ctq
    imageTag: 15.1.x.x.x
    quantum_db_password: password
    ctqCfg: /oms/vertex/64bit/cfg
    ctqCfgName: CTQ Test
    ctqSmObj: ./dm_vertex_ctq30206.so
```

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

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace BrmNameSpace
```

   where:

   - *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
   - *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

Deploying with Vertex Sales Tax Q Series

You deploy Vertex Manager with Vertex STQ by doing the following:

1. Copying the required libraries from the Vertex STQ installation directory to your **$PIN_HOME/req_libs** directory.

2. Layer the default images provided by Oracle. For example, to layer dm-vertex with Vertex STQ, you could add these sample commands to its Dockerfile:

   ```
   FROM dm_vertex:15.1.x.x.x

   USER root
   COPY ["req_libs/libvst*.so", "req_libs/libqutil*.so", "req_libs/libloc*.so",
   "/oms/lib/"]
   RUN chown omsuser:root -R /oms/lib/ /lib64
   USER omsuser
   ```

3. Build your new Vertex Manager image. For example:

   ```
   podman build --format docker --tag dm_vertex_stq:15.1.x.x.x --file
   Dockerfile_vertex_stq .
   ```

4. Update these key in your **override-values.yaml** file for **oc-cn-helm-chart**:

   ```
   dm_vertex:
     isEnabled: true
     deployment:
       replicaCount: 1
       imageName: dm_vertex
       imageTag: 15.1.x.x.x
       quantum_db_password: password
   ```

5. Update these entries in your **oc-cn-helm-chart/templates/ configmap_env_dm_vertex.yaml** file:

   ```
   SERVICE_FQDN: dm-vertex
   QUANTUM_DB_SOURCE: quantum
   QUANTUM_DB_SERVER: qsu122a
   QUANTUM_DB_USER: quantum
   ```

6. Update these entries in your **oc-cn-helm-chart/templates/ configmap_odbc_ini_dm_vertex.yaml** file:

   ```
   data:
     odbc.ini: |
       [ODBC Data Sources]
       Server = Oracle Server v12.2
       [Server]
       Description = Oracle Server v12.2
       Driver = /usr/lib/oracle/19.20/client64/lib/libsqora.so.19.1
       Servername = PINDB
       UserID = DBuser
       Password = DBpassword
       Port = 1521
       Trace = yes
       TraceFile = /oms_logs/odbc.log
       Database = //DBhostname:DBport
   ```

   where:

   - *Server* is the name of the server on which the Vertex database is installed.
   - *DBuser* is the Vertex database schema user name.
   - *DBpassword* is the password for the Vertex database schema user.
   - *DBhostname* is the host name of the machine on which the Vertex tax calculation database is installed.
   - *DBport* is the port number of the Vertex tax calculation database.

7. Set these entries in your **oc-cn-helm-chart/templates/ configmap_pin_conf_dm_vertex.yaml** file:

```
- dm_vertex quantum_sm_obj ./dm_vertex_stq100.so
- dm_vertex quantumdb_source ${QUANTUM_DB_SOURCE}
- dm_vertex quantumdb_server ${QUANTUM_DB_SERVER}
- dm_vertex quantumdb_user ${QUANTUM_DB_USER}
```

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

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

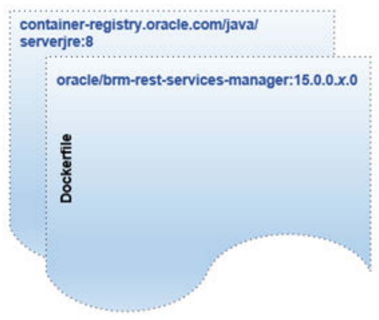
   where:

   - *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
   - *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **values.yaml** file for **oc-cn-helm-chart**.

Building BRM REST Services Manager Images

In a production deployment, containers for BRM REST Services Manager will run in their own pods on a Kubernetes node. Figure 15-1 shows how to stack images for BRM REST Services Manager.

Figure 15-1 Image Stack for BRM REST Services Manager


container-registry.oracle.com/java/serverjre:8
oracle/brm-rest-services-manager:15.0.0.x.0
Dockerfile

In this figure:

- **container-registry.oracle.com/java/serverjre:8**: The base image on which BRM REST Services Manager will be deployed. The official image is available at https://container-registry.oracle.com/.
- **oracle/brm-rest-services-manager:15.1.*x.x.x***: The sample Dockerfile and related scripts used for creating the BRM REST Services Manager image (**oracle/brm-rest-services-manager:15.1.*x.x.x***).

The **oc-cn-docker-files/ocrsm/brm_rest_services_manager** directory in the **oc-cn-docker-files-15.1.*x.x.x*.tgz** package contains a Dockerfile, container scripts, and an API JAR file.

You can load or build the BRM REST Services Manager image in the following ways:

- The **oc-cn-brm-rest-services-manager-15.1.*x.x.x*.tar** image is included in the package. Apply the image in your machine by running this command:

  ```
  podman load < oc-cn-brm-rest-services-manager-15.1.x.x.x.tar
  ```

- If the image needs customization, modify the Dockerfile and then deploy it using this command:

  ```
  podman build --format docker --tag oracle/brm-rest-services-manager:15.1.x.x.x .
  ```

Building PDC REST Services Manager Images

In a production deployment, containers for PDC REST Services Manager will run in their own pods on a Kubernetes node. You create PDC REST Services Manager images by stacking these Dockerfiles in the following order:

1. **container-registry.oracle.com/java/serverjre:8**: The base image on which PDC REST Services Manager will be deployed. The official image is available at https://container-registry.oracle.com/.
2. **oracle/pdcrsm:15.1.**_x.x.x_: The sample Dockerfile and related scripts used for creating the PDC REST Services Manager image (**oracle/pdcrsm:15.1.**_x.x.x_).

To build PDC REST Services Manager images:

1. Copy the Dockerfile and the **oc-cn-pdc-rsm-jars-15.1.**_x.x.x_ file into the current working directory.
2. Run the following commands:

   ```
   tar xvf oc-cn-pdc-rsm-jars-15.1.x.x.x.tar
   podman build --format docker --tag oracle/pdcrsm:15.1.x.x.x .
   ```

Building PDC Images

To build the PDC image:

1. Download **PricingDesignCenter-15.1.**_x.x.x_**.zip** to the _ParentFolder_**/Docker_files/PDCImage/other-files** directory.
2. Pull the Java Image from the Oracle Container Registry (https://container-registry.oracle.com). This image is regularly updated with the latest security fixes. You can pull this image to your local system, where you will build other images, with the name **container-registry.oracle.com/java/serverjre:**_JavaVersion_.

   where _JavaVersion_ is the Oracle Java version number. See _BRM Compatibility Matrix_ for supported versions.
3. Set the following environment variables:
   - HTTP_PROXY: Set this to the host name or IP address of your proxy server
   - JAVA_VERSION: Set this to **container-registry.oracle.com/java/serverjre:**_JavaVersion_
   - BRM_VERSION: Set this to **15.1.**_x.x.x_
4. Build your Oracle PDC BRM integration image by entering this command from the _ParentFolder_**/Docker_files/PDCImage** directory:

   ```
   podman build --format docker --force-rm=true --no-cache=true --build-arg
   HTTP_PROXY=$HTTP_PROXY --build-arg JAVA_VERSION=$JAVA_VERSION --tag $IMAGE_NAME --
   file Dockerfile .
   ```
5. (Optional) To use custom fields in your PDC RUM expressions:
   a. Create a **custom_flds.h** file that contains your custom fields.

For information about the syntax to use in a header file, view the *BRM_home***/include/pin_flds.h** file in the brm-sdk pod.

b. Parse the **custom_flds.h** file and generate a **custom_flds.bat** file.

c. Layer the *BRM_home***/lib/custom_flds.bat** file in the following images: brm-apps, pdc, and cm.

   For example, to layer the file in the pdc image:

```
FROM pdc:15.1.x.x.x

USER root
COPY custom_flds.bat ${PIN_HOME}/lib/custom_flds.bat
RUN chown=oracle:root ${PIN_HOME}/lib/custom_flds.bat
```

d. In your brm-apps-2 ConfigMap (**configmap_pin_conf_brm_apps_2.yaml**), add the following entry under the **load_config.conf** section:

```
load_config.conf: |
# Making custom fields entry
   - - ops_fields_extension_file ${PIN_HOME}/lib/custom_flds.h
```

e. In your CM ConfigMap (**configmap_pin_conf_cm.yaml**), add the following entry under the **pin.conf** section:

```
data:
   pin.conf:
   # Making custom fields entry
   - - ops_fields_extension_file ${PIN_HOME}/lib/custom_flds.h
```

f. In your **testnap** ConfigMap (**configmap_pin_conf_testnap**), add the following entry under the **pin.conf** section:

```
data:
   pin.conf:  |
   # Making custom fields entry
   - - ops_fields_extension_file ${PIN_HOME}/lib/custom_flds.h
```

g. In your **override-values.yaml** file for **oc-cn-helm-chart**, update the **imageTag** keys to point to the new cm, brm-apps and pdc images.

   Note:

   Skip this step if your **override-values.yaml** file does not already contain **imageTag** keys for the cm, brm-apps, and pdc images.

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

```
helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace BrmNameSpace
```

Building Pipeline Configuration Center Images

The Pipeline Configuration Center image extends the Fusion Middleware Infrastructure image by packaging its own installer **PipelineConfigurationCenter_15.1.***x.x.x***_generic.jar** file along with scripts and configurations.

To build your own image of Pipeline Configuration Center, you must have these base images ready. The **oc-cn-docker-files-15.1.***x.x.x***.tgz** package includes references to all Dockerfiles and scripts that are needed to build images of Pipeline Configuration Center. You can refer to them when building a Pipeline Configuration Center image in your own environment.

Pulling the Fusion Middleware Infrastructure Image

The Fusion Middleware Infrastructure Image is available on the Oracle Container Registry (https://container-registry.oracle.com). This image is regularly updated with the latest security fixes. You can pull this image to your local system, where you will build other images, with the name **container-registry.oracle.com/middleware/fmw-infrastructure_cpu:12.2.1.4-jdk8-ol7**.

Building the Pipeline Configuration Center Image

To build the Pipeline Configuration Center image, do this:

1. Go to the **oc-cn-docker-files/ocpcc/pcc** directory.
2. Download the Oracle Communications Pipeline Configuration Center installation JAR file.
3. Copy **PipelineConfigurationCenter_15.1.*x*.*x*.*x*_generic.jar** to the current working directory (**oc-cn-docker-files/ocpcc/pcc**).
4. Build the Pipeline Configuration Center image by entering this command:

```
podman build --format docker --tag oracle/pcc:15.1.x.x.x .
```

Building Billing Care Images

The Billing Care image extends the Linux image by packaging the application archive along with scripts and configurations.

To build your own image of Billing Care, you need the Linux and JRE images, available on the Oracle Container Registry (https://container-registry.oracle.com). These images are regularly updated with the latest security fixes. You can pull these images to your local system, where you will build other images, with the names:

- container-registry.oracle.com/os/oraclelinux:8
- container-registry.oracle.com/java/serverjre:8-oraclelinux8

The **oc-cn-docker-files-15.1.*x*.*x*.*x*.tgz** package includes references to all Dockerfiles and scripts that are needed to build images of Billing Care. You can refer to them when building a Billing Care image in your own environment.

Pulling the Fusion Middleware Infrastructure Image

The Fusion Middleware Infrastructure Image is available on the Oracle Container Registry (https://container-registry.oracle.com). This image is regularly updated with the latest security fixes. You can pull this image to your local system, where you will build other images, with the name **container-registry.oracle.com/middleware/fmw-infrastructure_cpu:12.2.1.4-jdk8-ol7**.

Building the Billing Care Image

To build the Billing Care image, do this:

1. Go to the **oc-cn-docker-files/ocbc/billing_care** directory.

2. Download the Oracle Communications Billing Care installation JAR file.
3. Copy **BillingCare_generic.jar** to the current working directory (**oc-cn-docker-files/ocbc/billing_care**).
4. Build the Billing Care image by entering this command:

```
podman build --format docker --tag oracle/billingcare:15.1.x.x.x .
```

Building Business Operations Center Images

The Business Operations Center image extends the Linux image by packaging the application archive along with scripts and configurations.

To build your own image of Business Operations Center, you need the Linux and JRE images, available on the Oracle Container Registry (https://container-registry.oracle.com). These images are regularly updated with the latest security fixes. You can pull these images to your local system, where you will build other images, with the names:

- container-registry.oracle.com/os/oraclelinux:8
- container-registry.oracle.com/java/serverjre:8-oraclelinux8

The **oc-cn-docker-files-15.1.*x*.*x*.*x*.tgz** package includes references to all of the Dockerfiles and scripts needed to build images of Business Operations Center. You can refer to them when building a Business Operations Center image in your own environment.

To build the Business Operations Center image, do this:

1. Go to the **oc-cn-docker-files/ocboc/boc** directory.
2. Download the Oracle Communications Business Operations Center installation JAR file.
3. Copy **BusinessOperationsCenter_generic.jar** to the current working directory (**oc-cn-docker-files/ocboc/boc**).
4. Build the Business Operations Center image by entering this command:

```
podman build --format docker --tag oracle/boc:15.1.x.x.x .
```

Part V

Upgrading BRM Cloud Native

This part provides information about upgrading your Oracle Communications Billing and Revenue Management (BRM) cloud native environment to the latest patch set or interim patch release. It contains the following chapters:

16 Upgrading Your BRM Cloud Native Environment

Learn how to upgrade your Oracle Communications Billing and Revenue Management (BRM) cloud native environment to the latest release.

Topics in this document:

In this document, the BRM release currently running on your production system is called the *old* release. The release you are upgrading to is called the *new* release. For example, if you upgrade from BRM 12.0.0.*x*.0 to BRM 15.1, 12.0.0.*x*.0 is the old release and 15.1 is the new one.

Tasks for the BRM Cloud Native Upgrade

This section provides a list of tasks required to upgrade your BRM cloud native deployment to the latest release, patch set, or interim patch release. All patch sets and interim patches are cumulative, so they include the fixes from previous patch sets and interim patches. You can perform a direct upgrade from one patch set to another. For example, you can perform a direct upgrade of BRM cloud native from 12.0.0.*x*.0 to 15.1.

To upgrade your BRM cloud native deployment, complete these tasks in the specified order:

1. If you are upgrading from 12.0.0.2.0 to 12.0.0.3.0 or later, migrate your BRM cloud native Helm charts to the v3.*x* format by using the **helm2to3** utility. The Helm charts in BRM cloud native 12.0.0.2.0 use Helm v2.*x*, and later patch set releases use Helm v3.*x*. Helm v3.*x* doesn't readily understand the releases created by Helm v2.*x*.

   For more information, see "Migrating Helm v2 to v3" in the Helm documentation. The documentation contains references to the migration plugin and to a blog with a comprehensive walk-through of steps using a sample chart.
2. Upgrade your BRM cloud native database schema. See "Upgrading Your Database Schema".
3. Upgrade your BRM cloud native services. See "Upgrading Your BRM Cloud Native Services".
4. Upgrade your Elastic Charging Engine (ECE) cloud native services. See "Upgrading Your ECE Cloud Native Services".
5. Upgrade your client application services in any order:
   - Upgrade your Pricing Design Center (PDC) service and database schema. See "Upgrading Your PDC Cloud Native Services".
   - Upgrade your BRM REST Services Manager service. See "Upgrading BRM REST Services Manager".
   - Upgrade your Business Operations Center service and database schema. See "Upgrading Your Business Operations Center Cloud Native Services".

- Upgrade your Pipeline Configuration Center (PCC) service. See "Upgrading Your PCC Cloud Native Services".
- Upgrade your Billing Care and Billing Care REST API services. See "Upgrading Your Billing Care and Billing Care REST API Cloud Native Services".

Upgrading Your Database Schema

To upgrade your BRM cloud native database schema to the 15.1 release:

1. Download Oracle Communications Cloud Native Database Initializer Helm Chart 15.1 from the Oracle Software Delivery Cloud website. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and load the BRM 15.1 cloud native component images in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. Extract the BRM database initializer Helm chart from the archive. For example:

   ```
   tar xvzf oc-cn-init-db-helm-chart-15.1.x.x.x.tgz
   ```

   If you are extracting an interim patch, the file name will also have the interim patch number appended to it, such as **oc-cn-init-db-helm-chart-15.1.0.0.0-12345678.tgz**.
4. Copy the **/oms/wallet/client** files (**ewallet.p12** and **cwallet.sso**) from the BRM 12.0.0.*x*.0 version of the dm-oracle pod to the BRM 15.1 **oc-cn-init-db-helm-chart/existing_wallet/** directory.

   Note:

   This step is only required if you are upgrading from 12.0.0.*x*.0 to 15.1. Skip this step if you are upgrading from 15.0.*x*.0.0 to 15.1.*x*.*x*.*x*.
5. Create an **override-init-db-15.yaml** file for the 15.1 version of **oc-cn-init-db-helm-chart**.
6. In the **override-init-db-15.yaml** file, do the following:
   - Set the **ocbrm.is_upgrade** key to **true**.
   - Set the **existing_rootkey_wallet** key to **true**.
   - Set the other keys in Table 6-1 as needed.

   Note:

   The BRM root password, wallet passwords, and database details should be the same as in your old release.
7. Validate the **oc-cn-init-db-helm-chart** 15.1 chart's content by using the **helm lint** command.
   - For Helm 3.6.0 and later releases, enter this command from the **helmcharts** directory:

```
helm lint --strict oc-cn-init-db-helm-chart --values oc-cn-init-db-helm-chart/
values.yaml --values override-init-db-15.yaml
```

- For previous Helm releases, enter this command from the **helmcharts** directory:

```
helm lint --strict oc-cn-init-db-helm-chart
```

You'll see this if the command completes successfully:

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

8. Upgrade the database schema by entering this command from the **helmcharts** directory. Ensure that you run the Helm chart with a new release name and namespace.

```
helm install newRelease oc-cn-init-db-helm-chart --namespace newNameSpace --values
override-init-db-15.yaml
```

where:

- *newRelease* is the release name for your new 15.1 release. This release name must be different from that of your old release.
- *newNameSpace* is the namespace in which to create BRM Kubernetes objects for the new 15.1 release. This namespace must be different from that of your old release.

Your BRM cloud native database schema is upgraded to the new 15.1 release.

To determine if the upgrade was successful, enter the following:

```
kubectl --namespace newNameSpace get pods
```

If successful, you will see something similar to this:

```
NAME           READY   STATUS     RESTARTS   AGE
upgrade-wc6sx  0/1     Completed  0          22h
```

Upgrading Your BRM Cloud Native Services

Note:

The steps for upgrading your BRM cloud native services are the same for old and new schemas.

When you upgrade your BRM cloud native services, it upgrades all BRM core services in your BRM cloud native environment.

To upgrade your BRM cloud native services to the 15.1 release:

1. Download and install Oracle Communications Cloud Native Helm Chart 15.1 from the Oracle Software Delivery Cloud website. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and load the BRM 15.1 cloud native images in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. Extract the BRM 15.1 Helm chart from the archive:

```
tar xvzf oc-cn-helm-chart-15.1.x.x.x.tgz
```

If you are extracting an interim patch, the file name will also have the interim patch number appended to it, such as **oc-cn-helm-chart-15.1.*x*.*x*.*x*-12345678.tgz**.

4. Create an **upgrade-brm-15.yaml** file for the 15.1 version of **oc-cn-helm-chart**.
5. In the **upgrade-brm-15.yaml** file, set the following keys:
   - Set the **ocbrm.is_upgrade** key to **true**.
   - Set the **ocbrm.existing_rootkey_wallet** key to **false**.
   - Set the other keys in Table 7-4 as needed.
6. Run the **helm upgrade** command for the 15.1 version of **oc-cn-helm-chart** using the same release name and namespace that you used for your old release:

   ```
   helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues --values
   upgrade-brm-15.yaml --namespace oldBrmNamespace
   ```

   where:
   - *oldBrmReleaseName* is the release name assigned to your old **oc-cn-helm-chart** installation.
   - *oldOverrideValues* is the file name and path to the **override-values.yaml** file for your old BRM installation.
   - *oldBrmNamespace* is the same namespace as for your old BRM deployment.
7. If you are upgrading a multischema system, do the following:
   a. Add the following lines to the 15.1 version of the **oc-cn-helm-chart/brmapps_scripts/loadme.sh** script:

      ```
      #!/bin/sh

      cd /oms/setup/scripts; perl pin_multidb.pl
      cd /oms/setup/scripts; perl pin_amt_install.pl -m
      exit 0;
      ```

   b. Enable the brm-apps job by setting these keys in your **override-values.yaml** file for the 15.1 version of **oc-cn-helm-chart**:
      - **ocbrm.brm_apps.job.isEnabled**: Set this to **true**
      - **ocbrm.brm_apps.job.isMultiSchema**: Set this to **false**
   c. Run the **helm upgrade** command for **oc-cn-helm-chart**:

      ```
      helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --
      namespace BrmNameSpace
      ```

   d. Restart the amt pod.

   Your BRM core services have been upgraded to the latest 15.1 release.
8. After the upgrade completes successfully, restart your WebLogic Server Administration Server and Managed Servers by following the instructions in https://oracle.github.io/weblogic-kubernetes-operator/faq/domain-secret-mismatch/.

Note:

The first time you run **pin_virtual_time** after upgrading the BRM core services to the 15.1 release, it generates a new 64-bit **pin_virtual_time_file** utility. You must restart the CM after the 64-bit **pin_virtual_time_file** is created. To do so:

1. In the **override-values.yaml** for **oc-cn-helm-chart**, set the following keys:
   - **ocbrm.config_jobs.restart_count**: Increment the value by 1.
   - **ocbrm.config_jobs.run_apps**: Set this to **true**.
2. Run the **helm upgrade** command for **oc-cn-helm-chart** using the same release name and namespace that you used for your old release:

   ```
   helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues
   --values upgrade-brm.yaml --namespace oldBrmNamespace
   ```

Upgrading Your ECE Cloud Native Services

To upgrade your ECE cloud native services to the 15.1 release:

1. Download the ECE cloud native Helm chart. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Extract the ECE Helm chart from the archive into a separate staging area. For example:

   ```
   tar xvzf oc-cn-ece-helm-chart-15.1.x.x.x.tgz StagingArea
   ```

3. In your staging area, create a diff between the new **values.yaml** file and your old **oc-cn-ece-helm-chart/values.yaml** file. For example, if you are upgrading from 12.0.0.5.0 to 15.1, do a diff between the 12.0.0.5.0 and 15.1 versions of the **values.yaml** file.
4. Using the diff, make a list of the keys that were added, changed, and removed in the new release.
5. Open your old release's **override-values.yaml** file for **oc-cn-ece-helm-chart**. This file contains all of the customizations that you made in previous releases.
6. Do the following:
   - Add and configure any new keys that you want to use.
   - Delete the keys that were removed.
   - If a key's default value changed, determine whether you want to modify the key's value.
7. Ensure that you have at least three ecs and ecs1 pod replicas configured in the file:
   - **charging.ecs1.replicas**: Set this to a value of 3 or greater.
   - **charging.ecs.replicas**: Set this to a value of 3 or greater.
8. If your current ECE cloud native deployment does not support cdrstore tablespaces and you are upgrading to a version that requires them, do one of the following:
   - Continue without separate tablespaces for cdrstore. To do so, in your **override-values.yaml** file, set the **cdrstoretablespace** and **cdrstoreindexspace** keys to an empty value:

     ```
     cdrstoretablespace: ""
     cdrstoreindexspace: ""
     ```

- Use separate tablespaces for cdrstore. To do so, in **override-values.yaml** file, set the **cdrstoretablespace** and **cdrstoreindexspace** keys to the tablespace names.

  Also, grant quota on the new tablespaces to the ECE schema user. Connect to your database as the **system** user using SQL*Plus, and enter these commands:

  ```
  SQL> ALTER USER EceSchemaUser quota unlimited on ECECDRTABLESPACE;
  SQL> ALTER USER EceSchemaUser quota unlimited on ECECDRINDEXSPACE;
  ```

9. Save and close your **override-values.yaml** file to your staging area.
10. Delete all old ECE template files from your staging area's **oc-cn-ece-helm-chart/templates** directory.
11. Copy the new ECE template files to your staging area's **oc-cn-ece-helm-chart/templates** directory.
12. Upgrade your ECE cloud native services by running these commands:

    ```
    cd StagingArea/oc-cn-ece-helm-chart/
    sh upgradeECE_15.1.x.x.x.sh -o OverrideValuesFile -n BrmNameSpace -r EceReleaseName -
    s y
    ```

    where:

    - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **oc-cn-ece-helm-chart/values.yaml** file.
    - *BrmNameSpace* is the namespace in which BRM Kubernetes objects reside for the BRM Helm chart.
    - *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.

Upgrading ECE Cloud Native to the Latest Interim Patch

To upgrade your ECE cloud native services from 15.1 to the latest 15.1 interim patch:

1. Delete any existing ECE Kubernetes jobs:

   ```
   kubectl --namespace BrmNameSpace get job
   kubectl --namespace BrmNameSpace delete job JobName
   ```

   where *BrmNameSpace* is the namespace in which BRM Kubernetes objects reside for the BRM Helm chart, and *JobName* is the name of the Kubernetes job.
2. Download the latest 15.1 interim patch release from the My Oracle Support website (https://support.oracle.com).
3. Extract the interim patch's ECE Helm chart from the archive into a separate staging area. For example:

   ```
   tar xvzf oc-cn-ece-helm-chart-15.1.0.0.0.xyz.tgz StagingArea
   ```

   where *xyz* is the interim patch number.
4. In your staging area, create a diff between the 15.1 interim patch **values.yaml** file and your old 15.1 **oc-cn-ece-helm-chart/values.yaml** file.
5. Using the diff, make a list of the keys that were added, changed, and removed in the 15.1 interim patch release.
6. Open your 15.1 **override-values.yaml** file for **oc-cn-ece-helm-chart**.
7. Do the following:

- Add and configure any new keys that you want to use.
- Delete the keys that were removed.
- If a key's default value changed, determine whether you want to override the key's value.

8. Ensure that you have at least three ecs and ecs1 pod replicas configured in the file:
   - **charging.ecs1.replicas**: Set this to a value of 3 or greater.
   - **charging.ecs.replicas**: Set this to a value of 3 or greater.
9. Save and close your **override-values.yaml** file.
10. Upgrade your ECE cloud native services to the latest 15.1 interim patch release by running these commands:

```
cd StagingArea/oc-cn-ece-helm-chart/
sh upgradeECE_15.1.0.0.0.sh --override OverrideValuesFile --namespace BrmNameSpace --
release EceReleaseName
```

    where:
    - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **oc-cn-ece-helm-chart/values.yaml** file.
    - *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.

Upgrading Your PDC Cloud Native Services

Note:

- When you upgrade your PDC cloud native services, your PDC database is also upgraded.
- If you reuse an old PDC schema, you must choose the same rating engine as the old PDC installation. That is, if your old PDC installation uses Elastic Charging Engine (ECE) for usage rating, you cannot switch to the Real-time Rating and Batch Rating Engines. Likewise, you cannot switch to ECE if your old PDC installation uses the Real-time Rating and Batch Rating Engines for usage rating. If you attempt to switch rating engines, the PDC upgrade fails and generates an error message.
- You cannot migrate pricing data to PDC cloud native systems, because the PDC **MigrateBRMPricing** utility is not supported in cloud native environments.

To upgrade your PDC cloud native services and the PDC database to the latest 15.1 release:

1. Download and extract the BRM 15.1 cloud native Helm charts. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the PDC 15.1 component images into your repository in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".

- From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".

3. Remove all PDC pods, services, and volume mounts from your PDC 12.0.0.*x*.0 release by doing the following:
   a. In your **override-values.yaml** file for the 12.0.0.*x*.0 version of **oc-cn-helm-chart**, set the **ocpdc.isEnabled** key to **false**.
   b. Run the **helm upgrade** command for the 12.0.0.*x*.0 release of **oc-cn-helm-chart**:

      ```
      helm upgrade oldBrmReleaseName oc-cn-helm-chart --values
      oldBrmOverrideValuesFile --namespace oldBrmNameSpace
      ```

      where:

      - *oldBrmReleaseName* is the release name assigned to your 12.0.0.*x*.0 version of the **oc-cn-helm-chart** installation.
      - *oldBrmOverrideValuesFile* is the file name and path of your 12.0.0.*x*.0 version of the **override-values.yaml** file for **oc-cn-helm-chart**.
      - *oldBrmNameSpace* is the namespace for your 12.0.0.*x*.0 version of the BRM deployment.

      Note:

      Ensure that all PDC pods, services, and volume mounts have been deleted.

4. If you are upgrading from PDC 12.0.0.8.0 or later to PDC 15.1, do the following:
   a. In your **override-values.yaml** file for your *old release* of **oc-cn-op-job-helm-chart**, set the **ocpdc.isEnabled** key to **false**.
   b. Run the **helm upgrade** command for your *old release* of **oc-cn-op-job-helm-chart**:

      ```
      helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
      oldOpJobOverrideValuesFile --namespace oldBrmNameSpace
      ```

      where:

      - *oldOpJobReleaseName* is the release name assigned to your old release of the **oc-cn-op-job-helm-chart** installation.
      - *oldOpJobOverrideValuesFile* is the file name and path of your old release of the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.
      - *oldBrmNameSpace* is the namespace for your old BRM deployment.

      Ensure all PDC jobs are deleted.
   c. Compare your old release's **oc-cn-op-job-helm-chart/values.yaml** file with the 15.1 version of that file.
      - Create a diff between your old release's **values.yaml** file and the 15.1 release's **values.yaml** file.
      - Using the diff, make a list of the keys that were added, changed, and removed in the 15.1 release.
      - Open *oldOpJobOverrideValuesFile*. This file contains all of the customizations that you made in previous releases.
      - Do the following:

    - Add and configure any new keys that you want to use.
    - Delete the keys that were removed.
    - If a key's default value changed, determine whether you want to modify the key's value.
  - Close and save the file as **new-op-job-override-values.yaml**.

5. Do one of the following:
   - If you are upgrading from 12.0 Patch Set 8 or later, open your **new-op-job-override-values.yaml** file.
   - If you are upgrading from 12.0 Patch Set 7 or earlier, create an **override-values.yaml** file for the 15.1 version of **oc-cn-op-job-helm-chart** named **new-op-job-override-values.yaml**.
6. In **new-op-job-override-values.yaml**, set the following keys:
   - **ocpdc.isEnabled**: Set this to **true**.
   - **ocpdc.configEnv.upgrade**: Set this to **true**.
   - **ocpdc.configEnv.deployAndUpgradeSite2**: Set this to **false**.
7. Run the **helm upgrade** command for the 15.1 version of **oc-cn-op-job-helm-chart**:

   ```
   helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values new-op-job-
   override-values.yaml --namespace oldBrmNameSpace
   ```

   Wait for the PDC domain job to complete.
8. Compare your 12.0.0.*x*.0 release's **oc-cn-helm-chart/values.yaml** file with the 15.1 version of the file.
   - Create a diff between your old release's **values.yaml** file and the 15.1 release's **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the 15.1 release.
   - Open your 12.0.0.*x*.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     - Add and configure any new keys that you want to use.
     - Delete the keys that were removed.
     - If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the file as **new-brm-override-values.yaml**.
9. In **new-brm-override-values.yaml**, set the following keys:
   - **ocbrm.pdc_deployed**: Set this to **true**.
   - **ocpdc.configEnv.upgrade**: Set this to **true**.

     This value must match the one set in your **override-values.yaml** file for **oc-cn-op-job-helm-chart** (in Step 6).
   - **ocpdc.isEnabled**: Set this to **true**.

10. Run the **helm upgrade** command for the 15.1 version of **oc-cn-helm-chart**:

    ```
    helm upgrade oldBrmReleaseName oc-cn-helm-chart --values new-brm-override-
    values.yaml --namespace oldBrmNameSpace
    ```

Upgrading BRM REST Services Manager

To upgrade your BRM REST Services Manager cloud native services to the 15.1 release:

1. Download and extract the BRM cloud native Helm charts. See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the BRM REST Services Manager component image to your repository in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling WebLogic Images for PDC, Billing Care, Billing Care REST API, and Business Operations Center".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. Disable the brm-rest-services-manager service in your BRM cloud native environment.
   a. Create an **upgrade-brm-rsm.yaml** file. This file will be used by **oc-cn-helm-chart**.
   b. In your **upgrade-brm-rsm.yaml** file, set the **ocrsm.rsm.isEnabled** key to **false**.
   c. Stop the brm-rest-services-manager pod by running the Helm upgrade command for the **oc-cn-helm-chart**:

      ```
      helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues --
      values upgrade-brm-rsm.yaml --namespace oldBrmNamespace
      ```

      where:
      - *oldBrmReleaseName* is the BRM release name for your old release.
      - *oldOverrideValues* is the file name and path to the **override-values.yaml** file for your old **brm-rest-services-manager** installation.
      - *oldBrmNamespace* is the BRM namespace for your old release.
4. Wait for the brm-rest-services-manager pod to stop.
5. In your **upgrade-brm-rsm.yaml** file, set the these keys:
   - **ocrsm.rsm.isEnabled**: Set this to **true**.
   - **ocrsm.rsm.deployment.imageTag**: Set this to the new release number in the format 15.1.0.0.0 or 15.1.0.0.0-*nnnnnnnn* for interim patches.
6. Copy the SSL Certificate for BRM REST Services Manager.
   a. Create a directory named **rsm_keystore** under the newly extracted **oc-cn-helm-chart/rsm** directory.
   b. Copy the files created in the step "Generating an SSL Certificate for BRM REST Services Manager" to the newly created **oc-cn-helm-chart/rsm** directory.
   c. Start your brm-rest-services-manager services by running the Helm upgrade command for **oc-cn-helm-chart**:

      ```
      helm upgrade oldBRMReleaseName oc-cn-helm-chart --values oldOverrideValues --
      values upgrade-brm-rsm.yaml --namespace oldBrmNamespace
      ```

Upgrading Your Business Operations Center Cloud Native Services

The instructions to upgrade your Business Operations Center services differ depending on the patch set you are upgrading to or from.

- To upgrade the Business Operations Center service from 12.0.0.7.0 or earlier to 15.1, follow the instructions in "Upgrading Your Business Operations Center Cloud Native Service from 12.0.0.7.0 or Earlier to 15.1".
- To upgrade the Business Operations Center service from 12.0.0.8.0 to 15.1, follow the instructions in "Upgrading Your Business Operations Center Cloud Native Service from 12.0.0.8.0 to 15.1".

> **Note:**
>
> When you upgrade your Business Operations Center cloud native service, you can also upgrade your Business Operations Center database schema.

Upgrading Your Business Operations Center Cloud Native Service from 12.0.0.7.0 or Earlier to 15.1

To upgrade your Business Operations Center cloud native service and database schema from 12.0.0.7.0 or earlier to 15.1:

1. Download and extract the 15.1 versions of the BRM cloud native Helm chart (**oc-cn-helm-chart**) and the cloud native operator job chart (**oc-cn-op-job-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".

2. Download and push the Business Operations Center component image (**boc**) to your repository in one of these ways:

   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".

3. Disable all Business Operations Center 12.0.0.*x*.0 services in your BRM cloud native environment.

   a. Create an **upgrade-boc.yaml** file and then set the **ocboc.boc.isEnabled** key to **false**.

      The **upgrade-boc.yaml** file will be used with both **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

   b. Stop the WebLogic domain by running the **helm upgrade** command for **oc-cn-helm-chart**:

      ```
      helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues --
      values upgrade-boc.yaml --namespace oldBrmNamespace
      ```

      where:

      - *oldBrmReleaseName* is the BRM release name for your old release.
      - *oldOverrideValues* is the **override-values.yaml** file for your 12.0.0.*x*.0 release.

  - *oldBrmNamespace* is the BRM namespace for your old release.

c. Remove the WebLogic domain by running the **helm upgrade** command for **oc-cn-op-job-helm-chart**:

```
helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
oldOverrideValues --values upgrade-boc.yaml --namespace oldBrmNameSpace
```

   where *oldOpJobReleaseName* is the **oc-cn-op-job-helm-chart** release name for the old release.

4. Clean up the data in your Business Operations Center 12.0.0.*x*.0 persistent volumes (PVs).

   a. Clean up the domain home from the PV for Business Operations Center 12.0.0.*x*.0:

   ```
   rm -rf Domain_home/domains/domainUID
   ```

   where:

   - *Domain_home* is the location specified in the **ocboc.boc.wop.domainVolHostPath** key.
   - *domainUID* is the domain name specified in the **ocboc.boc.wop.domainUID** key. The default is **boc-domain**.

   See Table 9-1 for more information.

   b. Clean up the application home from the PV for Business Operations Center:

   ```
   rm -rf Application_home/BOC
   ```

   where *Application_home* is the path specified in the **ocboc.boc.wop.appVolHostPath** key.

5. Compare your old 12.0.0.*x*.0 versions of the **oc-cn-helm-chart/values.yaml** and **oc-cn-op-job-helm-chart/values.yaml** files with the 15.1 versions of those files.

   - Create a diff between the 15.1 **values.yaml** file and your old 12.0.0.*x*.0 **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the new release.
   - Open your old 12.0.0.*x*.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     – Add and configure any new keys that you want to use.
     – Delete the keys that were removed.
     – If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the files as *updatedoldOverrideValues*.

6. Deploy Business Operations Center 15.1 with the latest changes and upgrade the Business Operations Center database schema.

   a. In your **upgrade-boc.yaml** file, set these Business Operations Center keys:

   - **ocboc.boc.isEnabled**: Set this to **true**.
   - **ocboc.boc.deployment.imageTag**: Set this to the new release number in the format 15.1.0.*x*.0 for patch sets and 15.1.0.*x*.0-*nnnnnnnn* for interim patches.

   b. Run the **helm upgrade** command for **oc-cn-op-job-helm-chart**:

   ```
   helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
   updatedoldOverrideValues --values upgrade-boc.yaml --namespace oldBrmNameSpace
   ```

   Wait for the jobs to complete their tasks.

7. Start your Business Operations Center 15.1 services by running the **helm upgrade** command for **oc-cn-helm-chart**:

   ```
   helm upgrade oldBRMReleaseName oc-cn-helm-chart --values updatedoldOverrideValues --
   values upgrade-boc.yaml --namespace oldBrmNamespace
   ```

Upgrading Your Business Operations Center Cloud Native Service from 12.0.0.8.0 to 15.1

To upgrade your Business Operations Center cloud native service and database schema from 12.0.0.8.0 to 15.1:

1. Download and extract the 15.1 versions of the BRM cloud native Helm chart (**oc-cn-helm-chart**) and the cloud native operator job chart (**oc-cn-op-job-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".

2. Download and push the Business Operations Center 15.1 image (**boc**) to your repository in one of these ways:

   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".

3. Compare your old 12.0.0.8.0 versions of the **oc-cn-helm-chart/values.yaml** and **oc-cn-op-job-helm-chart/values.yaml** files with the 15.1 versions of those files.

   - Create a diff between the 15.1 **values.yaml** file and your old 12.0.0.8.0 **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the new release.
   - Open your old 12.0.0.8.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     – Add and configure any new keys that you want to use.
     – Delete the keys that were removed.
     – If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the files as *updatedOldOverrideValues*.

4. Deploy Business Operations Center 15.1 with the latest changes and upgrade the Business Operations Center database schema.

   a. In your **upgrade-boc.yaml** file, set the **ocboc.boc.deployment.app.imageTag** key to the new release number in the format 15.1.0.0.0 or 15.1.0.0.0-*nnnnnnnn* for interim patches.

   The **upgrade-boc.yaml** file will be used with both **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

b. Run the **helm upgrade** command for **oc-cn-op-job-helm-chart**:

```
helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
updatedOldOverrideValues --values upgrade-boc.yaml --namespace oldBrmNameSpace
```

   Wait for the jobs to complete their tasks.

5. Start your Business Operations Center 15.1 services by running the **helm upgrade** command for **oc-cn-helm-chart**:

```
helm upgrade oldBRMReleaseName oc-cn-helm-chart --values updatedOldOverrideValues --
values upgrade-boc.yaml --namespace oldBrmNamespace
```

Upgrading Your PCC Cloud Native Services

The instructions to upgrade your PCC cloud native services differ depending on the release you are upgrading from.

- To upgrade your PCC cloud native services from 12.0.0.*x*.0 or earlier to 15.0.0.0.0, follow the instructions in "Upgrading Your PCC Cloud Native Services from 12.0.0.*x*.0 or Earlier to 15.0.0.0.0".
- To upgrade your PCC cloud native services from 15.0 to 15.1, follow the instructions in "Upgrading Your PCC Cloud Native Services from 15.0 to 15.1".

Upgrading Your PCC Cloud Native Services from 12.0.0.*x*.0 or Earlier to 15.0.0.0.0

To upgrade your PCC cloud native services from 12.0.0.*x*.0 or earlier to the 15.0.0.0.0 release:

1. Download and extract the BRM 15.0.0.0.0 cloud native Helm chart (**oc-cn-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the PCC 15.0.0.0.0 component image (**oracle/pcc:15.0.0.0.0**) to your repository in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. Disable the PCC 12.0.0.*x*.0 cloud native services in your BRM cloud native environment:
   a. Create an **upgrade-pcc.yaml** file for **oc-cn-helm-chart**.
   b. In your **upgrade-pcc.yaml** file, set the **ocpcc.pcc.isEnabled** key to **false**.
   c. Stop the running pcc pod by running the **helm upgrade** command for the 12.0.0.*x*.0 version of **oc-cn-helm-chart**:

```
helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues --
values upgrade-pcc.yaml --namespace oldBrmNamespace
```

   where:
   - *oldBrmReleaseName* is the BRM release name for your 12.0.0.*x*.0 release.
   - *oldOverrideValues* is the file name and path to the **override-values.yaml** file for your 12.0.0.*x*.0 BRM installation.
   - *oldBrmNamespace* is the BRM namespace for your 12.0.0.*x*.0 release.
4. Wait for the pcc pod to stop.
5. Copy the SSL Certificate for PCC:

   a. Create a **keystore_pcc** directory under **oc-cn-helm-chart/pcc**.
   b. Copy the default PKCS12 certificate and KeyStore files to the **oc-cn-helm-chart/pcc** directory.

      During deployment, Helm uses the KeyStore files to create a Secret, which will be mounted as a volume inside the pcc pod.
   c. If your KeyStore files have file names different from what is specified in the **values.yaml** file, update the **keyStoreType**, **keyStoreIdentityFileName**, and **keyStoreTrustFileName** keys in your **override-values.yaml** file.

6. Compare your 12.0.0.*x*.0 release's **oc-cn-helm-chart/values.yaml** file with the 15.0.0.0.0 version of the file.
   - Create a diff between your 12.0.0.*x*.0 release's **values.yaml** file and the 15.0.0.0.0 release's **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the 15.0.0.0.0 release.
   - Open your 12.0.0.*x*.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     - Add and configure any new keys that you want to use.
     - Delete the keys that were removed.
     - If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the file as **new-brm-override-values.yaml**.
7. In your **upgrade-pcc.yaml** file, set the following keys:
   - **ocpcc.pcc.isEnabled**: Set this to **true**.
   - **ocpcc.pcc.deployment.imageTag**: Set this to the new release number in the format 15.0.0.0.0 or 15.0.0.0.0-*nnnnnnnn* for interim patches.
8. Start your PCC cloud native services by running the **helm upgrade** command for the 15.0.0.0.0 version of **oc-cn-helm-chart**:

```
helm upgrade oldBRMReleaseName oc-cn-helm-chart --values new-brm-override-
values.yaml --values upgrade-pcc.yaml --namespace oldBrmNamespace
```

Upgrading Your PCC Cloud Native Services from 15.0 to 15.1

To upgrade your PCC cloud native services from 15.0 to 15.1:

1. Download and extract the BRM 15.0 cloud native Helm charts (**oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the PCC 15.1 component image (**pcc**) to your repository in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
   - From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".
3. Disable all PCC 15.0 cloud native services in your BRM cloud native environment:

a. Create an **upgrade-pcc.yaml** file and then set the **ocpcc.pcc.isEnabled** key to **false**.

   The **upgrade-pcc.yaml** file will be used with both **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

b. Stop the WebLogic domain by running the **helm upgrade** command for the 15.0 version of **oc-cn-helm-chart**:

```
helm upgrade oldBrmReleaseName oc-cn-helm-chart --values
oldOverrideValues --values upgrade-pcc.yaml --namespace oldBrmNamespace
```

   where:

   - *oldBrmReleaseName* is the BRM release name for your 15.0 release.
   - *oldOverrideValues* is the **override-values.yaml** file for your 15.0 release.
   - *oldBrmNamespace* is the BRM namespace for your 15.0 release.

c. Remove the WebLogic domain by running the **helm upgrade** command for the 15.0 version of **oc-cn-op-job-helm-chart**:

```
helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
oldOverrideValues --values upgrade-pcc.yaml --namespace oldBrmNameSpace
```

   where *oldOpJobReleaseName* is the **oc-cn-op-job-helm-chart** release name for the 15.0 release.

4. Clean up the data in your PCC 15.0 persistent volumes (PVs).

   a. Clean up the domain home from the PV for PCC 15.0:

```
rm -rf Domain_home/domains/domainUID
```

   where:

   - *Domain_home* is the location specified in the **ocpcc.pcc.wop.domainVolHostPath** key.
   - *domainUID* is the domain name specified in the **ocpcc.pcc.wop.domainUID** key. The default is **pcc-domain**.

   b. Clean up the application home from the PV for PCC 15.0:

```
rm -rf Application_home/PCC
```

   where *Application_home* is the path specified in the o**cpcc.pcc.wop.appVolHostPath** key.

5. Compare your 15.0 version of the **oc-cn-helm-chart/values.yaml** file with the 15.1 version of that file.

   - Create a diff between the 15.1 **values.yaml** file and the 15.0 **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the 15.1 release.
   - Open your 15.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:

    - Add and configure any new keys that you want to use.
    - Delete the keys that were removed.
    - If a key's default value changed, determine whether you want to modify the key's value.
  - Close and save the file as **new-brm-override-values.yaml**.

6. Configure the 15.1 version of **oc-cn-op-job-helm-chart**:
   a. Create an **op-job-override-values.yaml** file.
   b. In the file, set the PCC-related keys for **oc-cn-op-job-helm-chart**. See "Adding Pipeline Configuration Center Keys for oc-cn-op-job-helm-chart".
7. In your **upgrade-pcc.yaml** file, set these PCC keys:
   - **ocpcc.pcc.isEnabled**: Set this to **true**.
   - **ocpcc.pcc.deployment.imageTag**: Set this to the new release number in the format 15.1.*x*.0.0 for patch sets and 15.1.*x*.0.0-*nnnnnnnn* for interim patches.
8. Deploy PCC 15.1 with the latest changes by running the **helm upgrade** command for **oc-cn-op-job-helm-chart**:

   ```
   helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values op-job-override-
   values.yaml --values upgrade-pcc.yaml --namespace oldBrmNameSpace
   ```
9. Start your PCC 15.1 cloud native services by running the **helm upgrade** command for **oc-cn-helm-chart**:

   ```
   helm upgrade oldBRMReleaseName oc-cn-helm-chart --values new-brm-override-
   values.yaml --values upgrade-pcc.yaml --namespace oldBrmNamespace
   ```

Upgrading Your Billing Care and Billing Care REST API Cloud Native Services

The instructions to use to upgrade your Billing Care and Billing Care REST API services are different, depending on the patch set you are upgrading from.

- To upgrade Billing Care and Billing Care REST API services from 12.0 Patch Set 7 or earlier, follow the instructions in "Upgrading Your Billing Care and Billing Care REST API Cloud Native Services from 12.0.0.7.0 or Earlier to 15.1".
- To upgrade the Billing Care and Billing Care REST API services from 12.0 Patch Set 8 or later, follow the instructions in "Upgrading Your Billing Care and Billing Care REST API Cloud Native Services from 12.0.0.8.0 to 15.1".

Upgrading Your Billing Care and Billing Care REST API Cloud Native Services from 12.0.0.7.0 or Earlier to 15.1

To upgrade Billing Care and Billing Care REST API from 12.0.0.7.0 or earlier to 15.1:

1. Download and extract the 15.1 versions of the BRM cloud native Helm chart (**oc-cn-helm-chart**) and the cloud native operator job chart (**oc-cn-op-job-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the 15.1 versions of the Billing Care image (**billingcare**) and the Billing Care REST API image (**bcws**) to your repository in one of these ways:

- From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".
- From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".

3. Disable all Billing Care and Billing Care REST API services in your BRM 12.0.0.*x*.0 cloud native environment.
   a. Create an **upgrade-billing.yaml** file.

      This file will be used with both **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.
   b. In your **upgrade-billing.yaml** file, set these keys:
      - **ocbc.bc.isEnabled**: Set this to **false**.
      - **ocbc.bcws.isEnabled**: Set this to **false**.
   c. Stop the WebLogic domain by running the Helm upgrade command for **oc-cn-helm-chart**:

      ```
      helm upgrade oldBrmReleaseName oc-cn-helm-chart --values oldOverrideValues --
      values upgrade-billing.yaml --namespace oldBrmNamespace
      ```

      where:
      - *oldBrmReleaseName* is the BRM release name for your old release.
      - *oldOverrideValues* is the file name and path to the **override-values.yaml** file for your old Billing Care installation.
      - *oldBrmNamespace* is the BRM namespace for your old release.
   d. Remove the WebLogic domain by running the Helm upgrade command for **oc-cn-op-job-helm-chart**:

      ```
      helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
      oldOverrideValues --values upgrade-billing.yaml --namespace oldBrmNameSpace
      ```

      where *oldOpJobReleaseName* is the **oc-cn-op-job-helm-chart** release name for the old release.
4. Clean up the data in the 12.0.0.*x*.0 versions of the Billing Care and Billing Care REST API persistent volumes (PVs).
   a. Clean up the domain home from the PV for Billing Care and Billing Care REST API:

      ```
      rm -rf Domain_home/domains/domainUID
      ```

      where:
      - *Domain_home* is the location specified in the **ocbc.bc.wop.domainVolHostPath** and **ocbc.bcws.wop.domainVolHostPath** keys.
      - *domainUID* is the domain name specified in the **ocbc.bc.wop.domainUID** and **ocbc.bcws.wop.domainUID** keys. The defaults are **billingcare-domain** and **bcws-domain**.

      See Table 9-3 and Table 9-5.
   b. Clean up the application home from the PV for Billing Care and Billing Care REST API:

      ```
      rm -rf Application_home/billingcare
      ```

      where *Application_home* is the path specified in the **ocbc.bc.wop.appVolHostPath** and **ocbc.bcws.wop.appVolHostPath** keys.

5. Compare your old 12.0.0.*x*.0 versions of the **oc-cn-helm-chart/values.yaml** and **oc-cn-op-job-helm-chart/values.yaml** files with the 15.1 versions of those files.
   - Create a diff between the 15.1 **values.yaml** file and your old 12.0.0.*x*.0 **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the new release.
   - Open your old 12.0.0.*x*.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     – Add and configure any new keys that you want to use.
     – Delete the keys that were removed.
     – If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the files as *updatedOldOverrideValues*.
6. In your **upgrade-billing.yaml** file, set these Billing Care and Billing Care REST API keys:
   - **ocbc.bc.isEnabled**: Set this to **true**.
   - **ocbc.bcws.isEnabled**: Set this to **true**.
   - **ocbc.bc.deployment.imageTag**: Set this to the new release number in the format 15.1.*x*.*x*.*x* or 15.1.*x*.*x*.*x*-*nnnnnnnn* for interim patches.
   - **ocbc.bcws.deployment.imageTag**: Set this to the new release number in the format 15.1.*x*.*x*.*x* or 15.1.*x*.*x*.*x*-*nnnnnnnn* for interim patches.
7. Deploy the 15.1 versions of the Billing Care and Billing Care REST API with the latest changes by running the Helm upgrade command for **oc-cn-op-job-helm-chart**:

```
helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
updatedOldOverrideValues --values upgrade-billing.yaml --namespace oldBrmNameSpace
```

   Wait for the jobs to complete their tasks.
8. Start your 15.1 versions of the Billing Care and Billing Care REST API services by running the Helm upgrade command for **oc-cn-helm-chart**:

```
helm upgrade oldBRMReleaseName oc-cn-helm-chart --values updatedOldOverrideValues --
values upgrade-billing.yaml --namespace oldBrmNamespace
```

Upgrading Your Billing Care and Billing Care REST API Cloud Native Services from 12.0.0.8.0 to 15.1

To upgrade Billing Care and Billing Care REST API cloud native from the 12.0.0.8.0 release to 15.1:

1. Download and extract the 15.1 versions of the BRM cloud native Helm chart (**oc-cn-helm-chart**) and the cloud native operator job chart (**oc-cn-op-job-helm-chart**).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files".
2. Download and push the 15.1 versions of the Billing Care image (**billingcare**) and the Billing Care REST API image (**bcws**) to your repository in one of these ways:
   - From the Oracle Container Registry. To do so, see "Pulling BRM Images from the Oracle Container Registry".

- From the Oracle Software Delivery website. To do so, see "Downloading BRM Images from Oracle Software Delivery Website".

3. Compare your old 12.0.0.8.0 versions of the **oc-cn-helm-chart/values.yaml** and **oc-cn-op-job-helm-chart/values.yaml** files with the 15.1 versions of those files.
   - Create a diff between the 15.1 **values.yaml** file and your old 12.0.0.8.0 **values.yaml** file.
   - Using the diff, make a list of the keys that were added, changed, and removed in the new release.
   - Open your old 12.0.0.8.0 release's **override-values.yaml** file. This file contains all of the customizations that you made in previous releases.
   - Do the following:
     – Add and configure any new keys that you want to use.
     – Delete the keys that were removed.
     – If a key's default value changed, determine whether you want to modify the key's value.
   - Close and save the files as *updatedOldOverrideValues*.
4. Create an **upgrade-billing.yaml** file and set these Billing Care and Billing Care REST API keys:
   - **ocbc.bc.deployment.app.imageTag**: Set this to the new release number in the format 15.1.*x*.*x*.*x* for patch sets and 15.1.*x*.*x*.*x*-*nnnnnnnn* for interim patches.
   - **ocbc.bcws.deployment.app.imageTag**: Set this to the new release number in the format 15.1.*x*.*x*.*x* for patch sets and 15.1.*x*.*x*.*x*-*nnnnnnnn* for interim patches.

   The **upgrade-billing.yaml** file will be used with both **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.
5. Deploy the 15.1 versions of Billing Care and Billing Care REST API with the latest changes by running the Helm upgrade command for **oc-cn-op-job-helm-chart**:

   ```
   helm upgrade oldOpJobReleaseName oc-cn-op-job-helm-chart --values
   updatedOldOverrideValues --values upgrade-billing.yaml --namespace oldBrmNameSpace
   ```

   Wait for the jobs to complete their tasks.
6. Start your 15.1 versions of the Billing Care and Billing Care REST API services by running the Helm upgrade command for **oc-cn-helm-chart**:

   ```
   helm upgrade oldBRMReleaseName oc-cn-helm-chart --values updatedOldOverrideValues --
   values upgrade-billing.yaml --namespace oldBrmNamespace
   ```

17

Performing Zero-Downtime Upgrades

Learn how to upgrade an Oracle Communications Billing and Revenue Management (BRM) cloud native deployment without having to take the environment offline. A zero-downtime upgrade allows your customers to continue using BRM's major services during the upgrade process.

Topics in this document:

- Performing a Zero-Downtime Upgrade of BRM
- Performing a Zero Downtime Upgrade of PDC

In this document, the BRM release running on your production system is called the *existing* release. The release you are upgrading to is called the *new* release. For example, if you upgrade from BRM 12.0 Patch Set 5 to BRM 15.1, 12.0 Patch Set 5 is the existing release and 15.1 is the new one.

Performing a Zero-Downtime Upgrade of BRM

You can perform a zero-downtime upgrade of your BRM cloud native services and the BRM database schema from 12.0.0.*x*.0 to 15.1.

To perform a zero-downtime upgrade of BRM cloud native:

1. Download the BRM 15.1 cloud native package from the Oracle Software Delivery website (https://support.oracle.com) or the Oracle Support website (https://support.oracle.com).
2. Configure and deploy the BRM 12.0.0.*x*.0 **oc-cn-helm-chart** Helm chart on your cloud native environment by doing the following:
   a. In your **override-values.yaml** file for the BRM 12.0.0.*x*.0 **oc-cn-helm-chart**, set the following keys:

   ```
   ocbrm:
      refreshInterval: 10
      terminationGracePeriodSeconds: 120
      pcpReconnectDelayOnSocketError: 10
      pcpConnectRetryDelayOnError: 10
   ```

   b. In your **oc-cn-helm-chart/templates/configmap_pin_conf_cm.yaml** file, set the following key:

   ```
   - cm pcm_connect_max_retries 10
   ```

   c. Ensure that at least two replica pods of cm, dm-oracle, dm-ifw-sync, and realtime-pipeline are up and running.
   d. (BRM 12.0.0.7.0 only) Apply BRM 12.0.0.7.0 Interim Patch 34939558 to your cloud native environment.
   e. (BRM 12.0.0.7.0 or later) In your BRM 12.0.0.*x*.0 **oc-cn-helm-chart/templates/configmap_pin_conf_dm_oracle.yaml** file, set the following key:

   ```
   - dm dm_ignore_fld_mismatch_err 1
   ```

f. Run the **helm upgrade** command for the 12.0.0.*x*.0 **oc-cn-helm-chart**:

```
helm upgrade Brm_12_ReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace Brm_12_NameSpace
```

   where:

   - *Brm_12_ReleaseName* is the release name assigned to your existing 12.0.0.*x*.0 **oc-cn-helm-chart** installation.
   - *OverrideValuesFile* is the file name and path to the file that overrides the **oc-cn-helm-chart/values.yaml** file.
   - *Brm_12_NameSpace* is the namespace for your existing 12.0.0.*x*.0 BRM deployment.

g. Back up your existing BRM 12.0.0.*x*.0 Helm charts.

h. Copy the 15.1 **oc-init-db-helm-chart** and **oc-cn-helm-chart** Helm charts to your BRM cloud native environment.

i. (12.0.0.6.0 or earlier) Upgrade only the dm-oracle pod from 12.0.0.*x*.0 to 15.1 by doing the following:

   i. In your **override-values.yaml** file for the 15.1 **oc-cn-helm-chart**, set the **ocbrm.dm_oracle.deployment.imageTag** key to **15.1.0.0.0**.

   ii. Run the **helm upgrade** command for the 15.1 **oc-cn-helm-chart**:

```
helm upgrade Brm_12_ReleaseName oc-cn-helm-chart --values OverrideValuesFile
--namespace Brm_12_NameSpace
```

   Verify that only the dm-oracle pod is running with the 15.1 image. The remaining pods run with 12.0.0.*x*.0 images.

3. Upgrade the BRM cloud native database schema to 15.1 by doing the following:

   a. Ensure that the **ConfigCacheRefreshInterval** business parameter in **bus_params_system.xml** is set to **0**.

   For information about how to set business parameters, see "Running Load Utilities through Configurator Jobs" in *BRM Cloud Native System Administrator's Guide*.

   b. In your **override-values.yaml** file for the 15.1 **oc-cn-init-db-helm-chart**, set the following keys:

```
ocbrm:
   is_upgrade: true
   existing_rootkey_wallet: true
```

   c. Copy the wallet files (**ewallet.p12** and **cwallet.sso**) from the **/oms/wallet/client/** directory of the 12.0.0.*x*.0 primary dm-oracle pod to the 15.1 **oc-cn-init-db-helm-chart/existing_wallet** directory.

   d. Upgrade the cloud native database to 15.1 using the **helm install** command:

```
helm install NewInitDbReleaseName oc-cn-init-db-helm-chart --values
OverrideValuesFile --namespace NewInitDbNameSpace
```

   where:

   - *NewInitDbReleaseName* is the release name assigned to your new 15.1 **oc-cn-init-db-helm-chart** installation.
   - *NewInitDbNameSpace* is the namespace for your new 15.1 **oc-cn-init-db-helm-chart** deployment.

Verify that the database has upgraded successfully.

e. (12.0.0.6.0 or earlier) Restart the 15.1 dm-oracle pod.

4. Upgrade the BRM cloud native server to 15.1 by doing the following:

> **Note:**
>
> To run billing during the BRM cloud native server upgrade for non-production systems, set the next billing date in the **oc-cn-helm-chart/templates/configmap_env_common.yaml** file to the following:
>
> ```
> VIRTUAL_TIME_SETTING: "-m 2 billingDate"
> ```
>
> For example, to set the next billing date to Feb 12, 2025, set *billingDate* to 021210002025.

a. Ensure the **ConfigCacheRefreshInterval** business parameter in **bus_params_system.xml** is set to **0**.

For information about how to set business parameters, see "Running Load Utilities through Configurator Jobs" in *BRM Cloud Native System Administrator's Guide*.

b. In your **override-values.yaml** file for the 15.1 **oc-cn-helm-chart**, set the following keys:

```
ocbrm:
   refreshInterval: 10
   terminationGracePeriodSeconds: 120
   pcpReconnectDelayOnSocketError: 10
   pcpConnectRetryDelayOnError: 10
   virtual_time:
      enabled: true
      sync_pvt_time: 5
```

c. In the 15.1 **oc-cn-helm-chart/templates/configmap_pin_conf_cm.yaml** file, set the following:

```
- cm pcm_connect_max_retries 10
- cm pcm_em_proto_vers 0
```

d. In the 15.1 **oc-cn-helm-chart/templates/configmap_pin_conf_rtp_pipeline.yaml** file, set the following:

```
- cm-em pcm_em_proto_vers 0
```

e. In the 15.1 **oc-cn-helm-chart/templates/cm.yaml** file, set the following:

```
spec.template.spec.containers(- name: cm).livenessProbe.initialDelaySeconds: 60
```

f. In the 15.1 **oc-cn-helm-chart/templates/configmap_pin_conf_dm_oracle.yaml** file, set the following:

```
- dm dm_ignore_fld_mismatch_err 1
```

g. Upgrade the BRM cloud native server to 15.1 using the **helm upgrade** command:

```
helm upgrade Brm_12_ReleaseName oc-cn-helm-chart --values OverrideValuesFile --
namespace Brm_12_NameSpace
```

Verify that all 12.0.0.*x*.0 pods terminate and all 15.1 pods come up and run with 15.1 images.

5. Revert the configuration values in your **override-values.yaml** file for the 15.1 **oc-cn-helm-chart**:

   a. In non-production systems, set the **VIRTUAL_TIME_SETTING** parameter to the default value in your **oc-cn-helm-chart/templates/configmap_env_common.yaml** file:

   ```
   VIRTUAL_TIME_SETTING: "-m 0"
   ```

   b. In your **oc-cn-helm-chart/templates/configmap_pin_conf_rtp_pipeline.yaml** file, remove or comment out the following entry:

   ```
   - cm-em pcm_em_proto_vers 0
   ```

   c. In your **oc-cn-helm-chart/templates/configmap_pin_conf_cm.yaml** file, remove or comment out the following entry:

   ```
   - cm pcm_em_proto_vers 0
   ```

   d. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **ocbrm.refreshInterval** key to its original value.

   e. Set the **ConfigCacheRefreshInterval** business parameter in **bus_params_system.xml** to its original value.

      For information about how to set business parameters, see "Running Load Utilities through Configurator Jobs" in *BRM Cloud Native System Administrator's Guide*.

   f. Run the **helm upgrade** command for the 15.1 **oc-cn-helm-chart**:

   ```
   helm upgrade Brm_12_ReleaseName oc-cn-helm-chart --values OverrideValuesFile --
   namespace Brm_12_NameSpace
   ```

Performing a Zero Downtime Upgrade of PDC

You can perform a zero downtime upgrade of your PDC cloud native services and the PDC database schema from the 12.0 or 12.0 Patch Set release to the 15.1 release.

You do so using a two-namespace approach in which you create an instance of PDC cloud native in a standby namespace, redirect PDC traffic to services in the standby namespace, upgrade PDC cloud native to release 15.1 in your original namespace, and then redirect PDC traffic back to your original namespace.

To upgrade PDC in zero downtime upgrade mode:

1. Create a temporary namespace, such as **BrmStandbyNameSpace**.
2. Clone your PDC *OverrideValuesFile* file to *StandbyOverrideValuesFile*.
3. In your *StandbyOverrideValuesFile* file for **oc-cn-op-job-helm-chart**, set the following keys:

```
ocpdc:
   configEnv:
      rcuPrefix: NewPrefix
      crossRefSchemaUserName: XrefSchema
      pdcSchemaUserName: PdcSchema
```

```
        deployAndUpgradeSite2: true
        upgrade: true
```

where:

- *NewPrefix* is the new prefix for the PDC domain RCU schema.
- *XrefSchema* is the same XREF schema name used for deploying PDC in **BrmNameSpace**.
- *PdcSchema* is the same PDC schema name used for deploying PDC in **BrmNameSpace**.

4. In your *StandbyOverrideValuesFile* file for **oc-cn-helm-chart**, set the following keys:

```
ocpdc:
    configEnv:
        upgrade: true
```

   These settings will upgrade the PDC and XREF schema.

5. Copy the following templates from your **BrmNameSpace** Helm chart **template** directory to the **BrmStandbyNameSpace** Helm chart **template** directory:

   - **secret_env_brm.yaml**
   - **configmap_pin_conf_brm_apps_2.yaml**
   - **configmap_loadifwconfig_reg.yaml**
   - **configmap_env_common.yaml**
   - **virtual_time_pvc.yaml**
   - **configmap_infranet_properties_brm_apps.yaml**
   - **config_jobs.yaml**
   - **storage_class_green.yaml**
   - **realtime_pipeline_common_pvc.yaml**
   - **configmap_tns_admin.yaml**
   - **secret_wallet_db.yaml**
   - **_helpers_utils.tpl**
   - **configmap_pdc_aux_engines.yaml**
   - **configmap_log_properties_pdc.yaml**
   - **configmap_env_pdc.yaml**
   - **configmap_env_pdc_rre.yaml**
   - **job_ie_pdc.yaml**
   - **domain_pdc.yaml**
   - **_pdchelpers.tpl**
   - **pdc_domain_monitoring_role.yaml**
   - **pdc_domain_monitoring_rbac.yaml**
   - **service_monitor_pdc_domain.yaml**
   - **secret_pdc.yaml**
   - **volume_pdc_brm.yaml**

- **deployment_pdc_rre.yaml**
- **deployment_pdc_bre.yaml**
- **deployment_pdc_syncpdc.yaml**

6. Create a configuration file named **cm-service-external-name.yaml** and add the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: cm
  namespace: BrmStandbyNameSpace
spec:
  externalName: cm.BrmNameSpace.svc.cluster.local
  internalTrafficPolicy: Cluster
  ports:
  - port: 11960
    protocol: TCP
    targetPort: 11960
  sessionAffinity: None
  type: ExternalName
```

7. Apply the configuration file to a resource:

```
kubectl apply -f cm-service-external-name.yaml
```

8. Deploy the PDC 12.0 Patch Set 7 or 8 Helm charts in your standby namespace:

```
helm install OpJobStandbyReleaseName oc-cn-op-job-helm-chart --values
StandbyOverrideValuesFile --namespace BrmStandbyNameSpace

helm install BrmStandbyReleaseName oc-cn-helm-chart --values
StandbyOverrideValuesFile --namespace BrmStandbyNameSpace
```

9. Redirect PDC traffic to services in **BrmStandbyNameSpace**.
10. Upgrade your PDC cloud native services to release 15.1 in **BrmNameSpace** while requests are temporarily routed to **BrmStandbyNameSpace**.
    a. Set the following keys in your **override-values.yaml** file for **oc-cn-op-job-helm-chart**:

    ```
    ocpdc:
       configEnv:
          deployAndUpgradeSite2: false
          upgrade: true
    ```

    b. Follow the instructions in "Upgrading Your PDC Cloud Native Services" to upgrade your original namespace to 15.1.
11. Redirect PDC traffic back to services in your original namespace (**BrmNameSpace**).

18

Performing Zero-Downtime Upgrades of Disaster Recovery Cloud Native Systems

You can perform a zero-downtime upgrade of an Oracle Communications Billing and Revenue Management (BRM) cloud native deployment in an active-active disaster recovery system.

Topics in this document:

- About the Zero-Downtime Upgrade of an Active-Active Disaster Recovery System
- Tasks for Upgrading a BRM Cloud Native Active-Active System

About the Zero-Downtime Upgrade of an Active-Active Disaster Recovery System

The steps for performing a zero-downtime cloud native upgrade of an active-active disaster recovery system assume that you are upgrading to the new release and assumes that your system contains a secondary BRM and ECE instance, which can be on a disaster recovery site or the primary site.

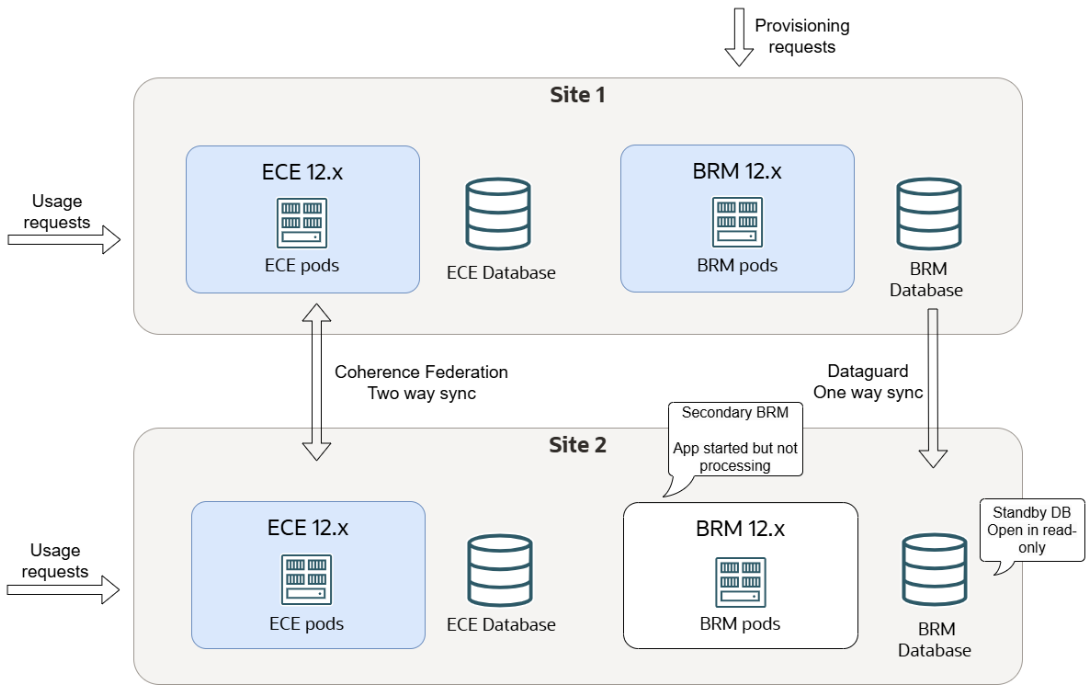
Caution:

Be aware that during the upgrade process:

- Usage processing is done in a single instance only.
- PDC is not available.
- ECE cache federation is supported from the old to the new version but not from the new to the old version.
- If an Oracle Data Guard role reversal is required for the BRM database, a short (few minutes) downtime occurs for provisioning.

Figure 18-1 shows initial state of a BRM cloud native active-active disaster recovery system.

Figure 18-1 Initial State of BRM Cloud Native Active-Active Disaster Recovery System



For information about the supported software versions, see *BRM Compatibility Matrix*.

Tasks for Upgrading a BRM Cloud Native Active-Active System

To upgrade your BRM cloud native active-active disaster recovery system using the zero-downtime upgrade process:

1. Turn off Site 2. See "Switching Off Site 2".
2. From Site 2, uninstall the old version of BRM and ECE. See "Uninstalling BRM and ECE from Site 2".
3. On Site 2, upgrade the Kubernetes platform and all prerequisite software to the versions required by the 15.*x* release. See *BRM Compatibility Matrix* for the list of supported software versions.

   For information, see "Upgrade a Cluster" in the Kubernetes documentation.
4. On Site 2, upgrade the BRM database schema to the 15.*x* release. See "Upgrading Your BRM Database Schema in Site 2".
5. On Site 2, install the BRM 15.*x* cloud native software. See "Installing BRM 15.*x* Cloud Native on Site 2".
6. From Site 2, drop the old persistence database schema. See "Dropping the ECE Persistence Database Schema from Site 2".
7. On Site 2, install ECE 15.*x*. See "Installing ECE 15.*x* Cloud Native on Site 2".
8. Transfer usage processing from Site 1 to Site 2. See "Failing Over Site 1 to Site 2".
9. From Site 1, uninstall the old version of BRM and ECE. See "Uninstalling BRM and ECE from Site 1".

10. On Site 1, upgrade the Kubernetes platform and all prerequisite software to the versions required by the 15.*x* release. See *BRM Compatibility Matrix* for the list of supported software versions.

    For information, see "Upgrade a Cluster" in the Kubernetes documentation.

11. From Site 1, drop the old ECE persistence database schema. See "Dropping the ECE Persistence Database Schema from Site 1".
12. On Site 1, install ECE 15.*x*. See "Installing ECE 15.*x* Cloud Native on Site 1".
13. Restart the federation process between Site 1 and Site 2. See "Federating ECE Cache Data Between Site 1 and Site 2".
14. If required, move the provisioning flow back to Site 1 and do an Oracle Data Guard role reversal.

> **Note:**
>
> The Data Guard role reversal may cause a few minutes of downtime in the provisioning flow, but it does not impact the usage flow.

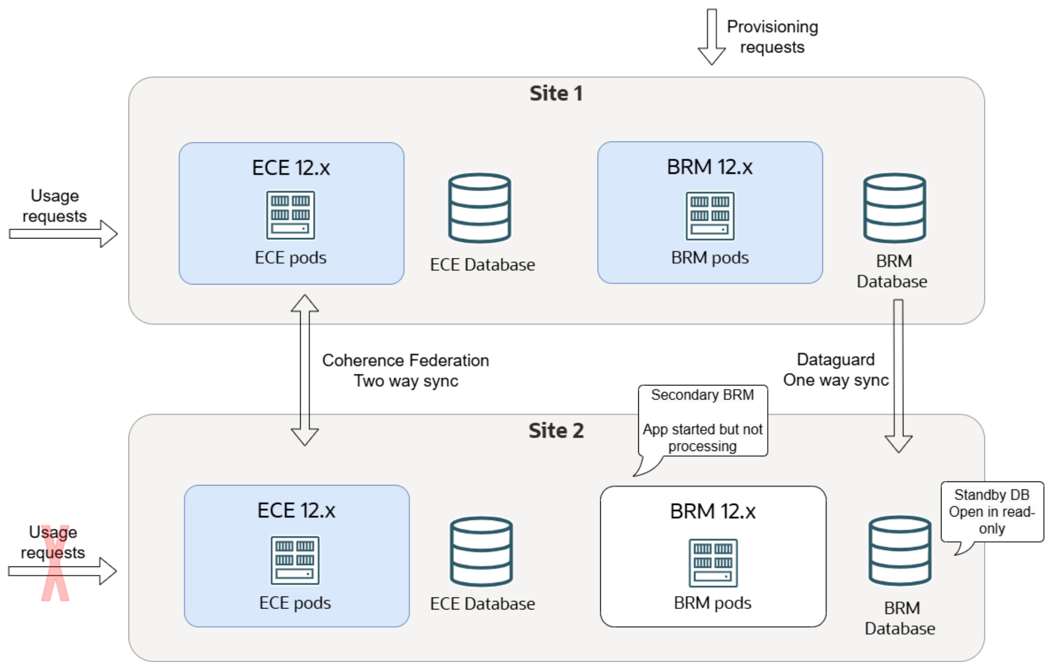
Switching Off Site 2

During the first phase of the upgrade process, Site 1 processes usage requests while you upgrade the software in Site 2.

To switch off Site 2:

1. Stop all usage requests to Site 2 and direct all usage requests to Site 1, as shown in Figure 18-2.

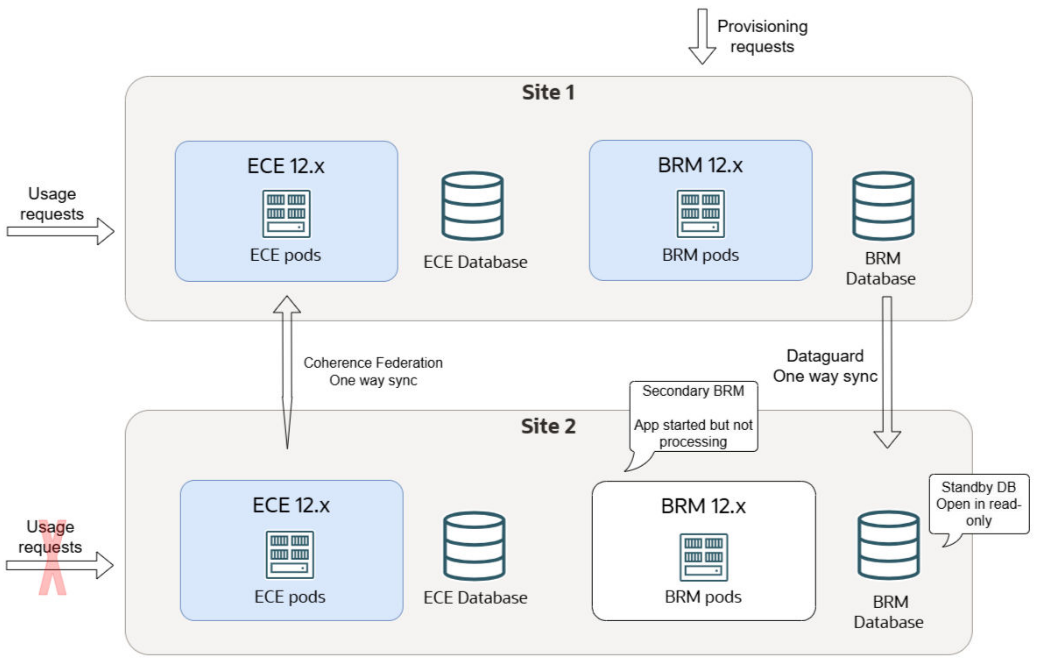
Figure 18-2 Usage Requests for Switching Off Site 2



2. Stop the connection from BRM on Site 1 to ECE on Site 2. To do so:
   - On ECE Site 2, stop the EM Gateway.
   - On BRM Site 1, remove any connection to the EM Gateway on Site 2.
3. On ECE Site 1, mark Site 2 as failed.

   This stops ECE from forwarding rating requests to Site 2 for subscribers with Site 2 as their preferred site. Usage requests are now rated on Site 1.
4. Disable the federation from ECE Site 1 to Site 2, as shown in Figure 18-3.

Figure 18-3 Disabled Federation from ECE Site 1 to Site 2



> **Note:**
>
> Keep the Site 2 to Site 1 federation active to drain any remaining data from Site 2.

5. Check that the federation backlog from Site 2 to Site 1 has been cleared.

   All traffic to Site 2 is stopped. Ensure all data from Site 2 has been synchronized with Site 1. The Coherence Federation Metrics should show as IDLE instead of YIELDING.
6. On Site 2, check that all rated events in the ECE cache have been extracted by running the following with the **query.sh** utility:

```
./query.sh
Coherence Command Line Tool
CohQl> select value() from AggregateObjectUsage
```

If successful, the query displays zero entries. See "query" in *ECE Implementing Charging* for information about the utility's syntax and parameters.

7. On Site 2, use SQL*Plus to check that all Site 2 rated events have been extracted from the persistence database and are present in BRM:

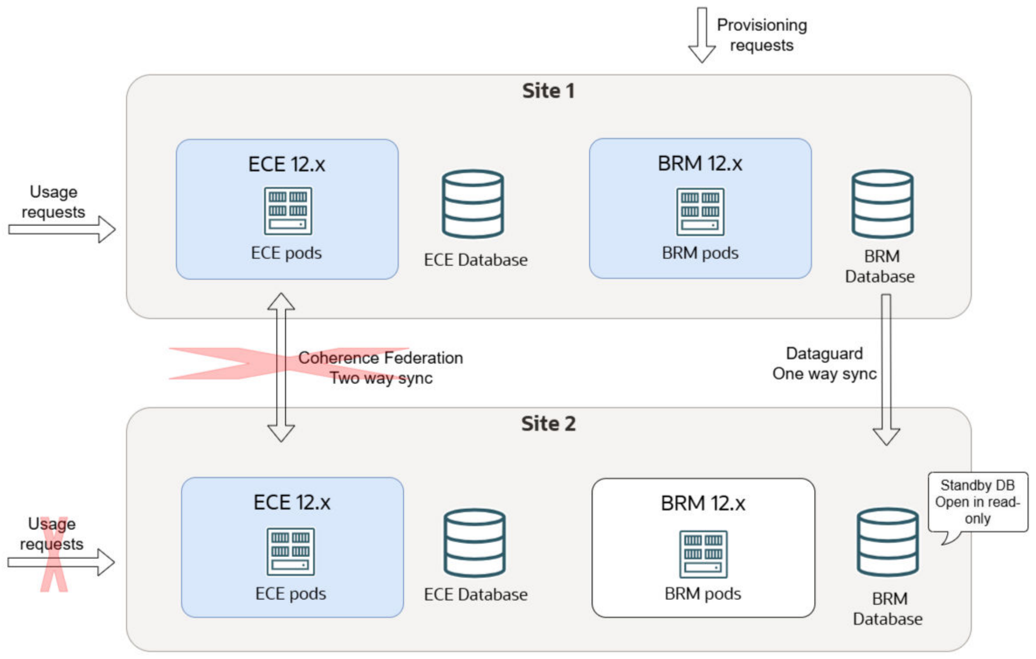
```
sqlplus pin@databaseName
Enter password: password

SQL> select count(*) from ratedevent_site2Name
```

where *databaseName* is the service name or database alias of the BRM database, and *password* is the password for the **pin** user.

> **Note:**
>
> The Site 2 persistence database might contain some Site 1 events. After the Site 2 to Site 1 federation process is stopped, these events are not extracted or purged. However, they will be processed on Site 1. You can ignore these events because they get purged when you re-create the Site 2 persistence database later.

8. Stop the federation process from Site 2 to Site 1, as shown in Figure 18-4.

Figure 18-4 Stopped Federation from ECE Site 2 to Site 1



All data is now synchronized with Site 1. Site 2 is isolated and ready for the upgrade.

Uninstalling BRM and ECE from Site 2

To uninstall the old version of BRM and ECE from Site 2:

1. Uninstall old version of BRM cloud native from Site 2:

```
helm uninstall BrmReleaseName --namespace BrmNameSpace
```

2. Uninstall the old version of ECE cloud native from Site 2:

```
helm uninstall EceReleaseName --namespace BrmNameSpace
```

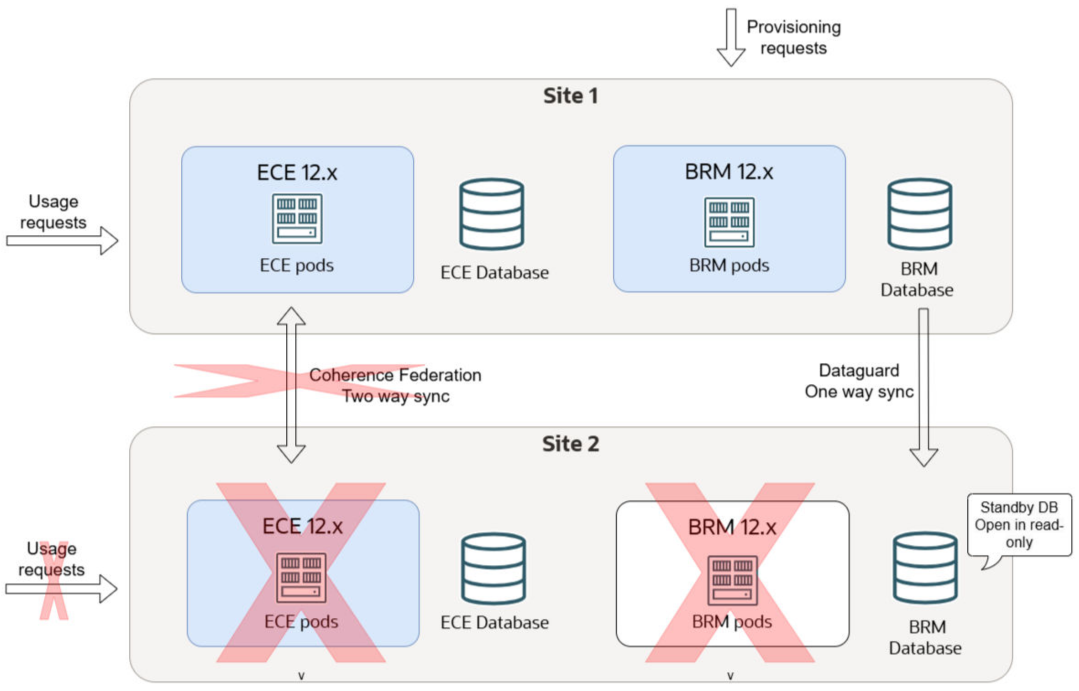
where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
- *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance.
- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.

These commands delete all the resources associated with the chart's last release and the release history.

Figure 18-5 shows uninstallation of BRM and ECE from Site 2.

Figure 18-5 Uninstallation of BRM and ECE from Site 2



Upgrading Your BRM Database Schema in Site 2

To upgrade your BRM database schema in Site 2 to release BRM 15.*x*:

1. Download and extract the BRM 15.*x* cloud native database initializer Helm chart (**oc-cn-init-db-helm-chart**) from Oracle Software Delivery Cloud (https://edelivery.oracle.com).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files" for more information.

2. Copy your old wallet files (**ewallet.p12** and **cwallet.sso**) from the dm-oracle pod's **/oms/wallet/client** directory to the BRM 15.*x* Helm chart's **oc-cn-init-db-helm-chart/existing_wallet/** directory.

   Note:

   Copying old wallet files is applicable only if the base version is 12.*x*.

3. Create an **override-init-db-15.yaml** file for the 15.*x* version of **oc-cn-init-db-helm-chart**.
4. In your **override-init-db-15.yaml** file, set the following keys:
   - **ocbrm.is_upgrade**: Set this to **true**.
   - **ocbrm.existing_rootkey_wallet**: Set this to **true**.
   - **ocbrm.wallet.client**: Set this to the password for the client wallet. This value must match that of your old release.
   - **ocbrm.wallet.server**: Set this to the password for the server wallet. This value must match that of your old release.
   - **ocbrm.wallet.root**: Set this to the password for the root wallet. This value must match that of your old release.
5. Enter this command from the **helmcharts** directory to upgrade the database schema. Ensure you run the 15.*x* **oc-cn-init-db-helm-chart** Helm chart with a new release name and namespace.

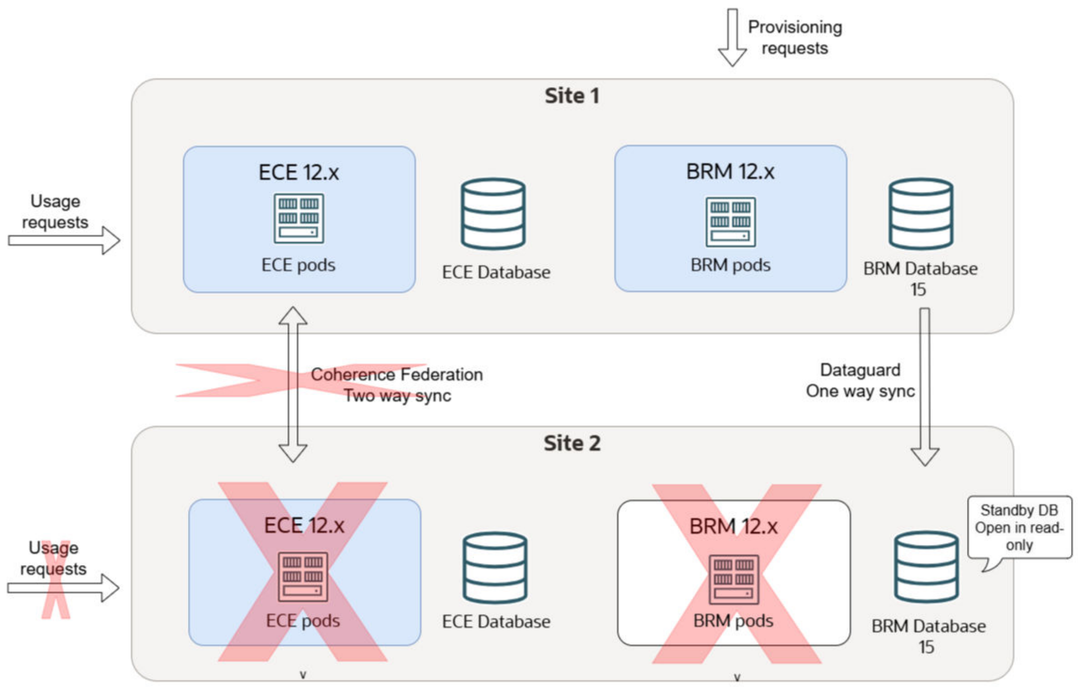
   ```
   helm install newInitDbReleaseName oc-cn-init-db-helm-chart --namespace
   newInitDbNameSpace --values oldOverrideValues --values override-init-
   db-15.yaml
   ```

   where:

   - *newInitDbReleaseName* is the new release name for the 15.*x* version of **oc-cn-init-db-helm-chart**.
   - *newInitDbNameSpace* is the new namespace for the 15.*x* version of **oc-cn-init-db-helm-chart**.
   - *oldOverrideValues* is the name and path to your old **override-values.yaml** file for **oc-cn-init-db-helm-cart**.

Figure 18-6 shows upgraded BRM database schema in Site 2.

Figure 18-6   Upgraded BRM Database Schema in Site 2



Installing BRM 15.*x* Cloud Native on Site 2

To configure and deploy BRM 15.*x* on your cloud native system on Site 2:

1. Download and extract the BRM 15.*x* cloud native (**oc-cn-helm-chart**) and BRM 15.*x* Operator Job Helm Chart (**oc-cn-op-job-helm-chart**) from Oracle Software Delivery Cloud (https://edelivery.oracle.com).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files" for more information.

2. To reuse your old SSL KeyStore with the new release, copy the PDC KeyStore files from the old Helm chart to the 15.*x* **oc-cn-op-job-helm-chart/pdc/pdc_keystore/** directory.

3. Create an **override-values-15.yaml** file.

   You will use this file with the 15.*x* version of **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

4. In your **override-values-15.yaml** file, set the following keys:

   - In the BRM section:
     – **ocbrm.is_upgrade**: Set this to **true**.
     – **ocbrm.existing_rootkey_wallet**: Set this to **false**.
     – **ocbrm.db.***: Set the BRM database schema details to the same values as your old release.
   - In the PDC section:
     – **ocpdc.configEnv.pdcSchemaUserName**: Set this to the same value as your old release.

   - **ocpdc.configEnv.crossRefSchemaUserName**: Set this to the same value as your old release.
   - **ocpdc.configEnv.rcuPrefix**: Set this key to a new prefix to create a new RCU schema.
   - **ocpdc.configEnv.transformation.upgrade**: Set this to **true**. (For upgrades to 15.0.0 only)
   - **ocpdc.secretValue.walletPassword**: Set this to the same value as your old release.
   - **ocpdc.configEnv.deployAndUpgradeSite2**: Set this to **true**. (For upgrades to 15.0.1 on later)
   - **ocpdc.configEnv.upgrade**: Set this to **true**. (For upgrades to 15.0.1 or later)
- In the Business Operations Center section:
   - **ocboc.boc.configEnv.bocSchemaUserName**: Set this to the same value as your old release.
   - **ocboc.boc.configEnv.runUpgrade**: Set this to **true**.
   - **ocboc.boc.configEnv.rcuPrefix**: Set this to a new prefix to create a new RCU schema.
   - **ocboc.boc.secretVal.***: Set the Business Operations Center passwords to the same values as your old release.
- In the Billing Care section:
   - **ocbc.bc.configEnv.rcuPrefix**: Set this to a new prefix to create a new RCU schema.
   - **ocbc.bc.secretVal.***: Set the Billing Care passwords to the same values as your old release.
- In the Billing Care REST API section:
   - **ocbc.bcws.configEnv.rcuPrefix**: Set this to a new prefix to create a new RCU schema.
   - **ocbc.bcws.secretVal.***: Set the Billing Care REST API passwords to the same values as your old release.

5. Create WebLogic domains by running the 15.*x* version of **oc-cn-op-job-helm-chart** from the **helmcharts** directory:

```
helm install oldOpJobReleaseName oc-cn-op-job-helm-chart --namespace
oldBrmNameSpace --namespace oldOverrideValuesFile --values override-
values-15.yaml
```

6. Install BRM cloud native services by running the 15.*x* version of **oc-cn-helm-chart** from the **helmcharts** directory:

```
helm install oldBrmReleaseName oc-cn-helm-chart --namespace
oldBrmNameSpace --values oldOverrideValuesFile --values override-
values-15.yaml
```

Dropping the ECE Persistence Database Schema from Site 2

To drop the ECE persistence database schema from Site 2:

1. Run the following command:

   ```
   DROP USER schemaname CASCADE
   ```

2. Uninstall the old version of the ECE Helm chart:

   ```
   helm uninstall oldEceReleaseName
   ```

   where *oldEceReleaseName* is the release name for your old version of **oc-cn-ece-helm-chart**.
3. Delete the old version of **ece-persistence-job** from your system by running this command:

   ```
   kubectl delete job ece-persistence-job --namespace oldBrmNameSpace
   ```

   where *oldBrmNameSpace* is the namespace for your old version of your BRM Helm release.

> **Note:**
>
> The ECE 15.*x* Helm chart re-creates the persistence database schema when you install it later.

Installing ECE 15.*x* Cloud Native on Site 2

To install ECE 15.*x* cloud native on Site 2:

1. Download and extract the ECE 15.*x* cloud native (**oc-cn-ece-helm-chart**) from Oracle Software Delivery Cloud (https://edelivery.oracle.com).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files" for more information.
2. Create an **override-values-ece-15.yaml** file.
3. In the file, configure your ECE 15.*x* cloud native services by following the instructions in "Configuring ECE Services".
4. Deploy the ECE 15.*x* cloud native services by entering this command from the **helmcharts** directory:

   ```
   helm install oldEceReleaseName oc-cn-ece-helm-chart --namespace
   oldBrmNameSpace --values oldOverrideValuesFile --values override-values-
   ece-15.yaml
   ```

   where:

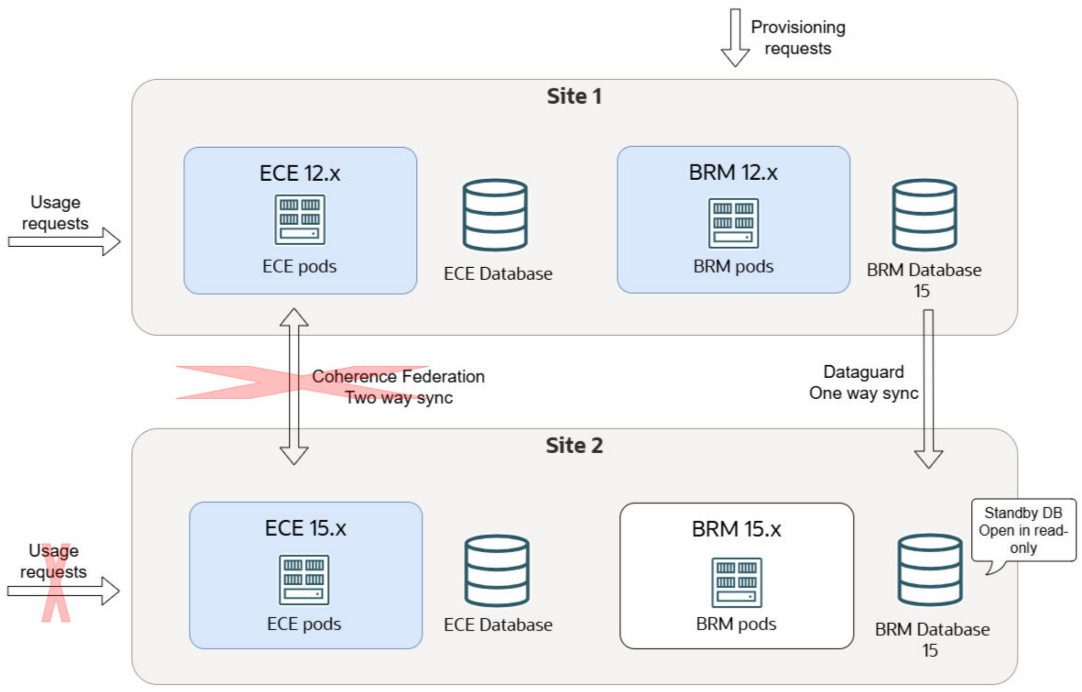
   - *oldEceReleaseName* is the release name for your old version of **oc-cn-ece-helm-chart**.
   - *oldBrmNameSpace* is the namespace for your old **oc-cn-helm-chart** deployment.
   - *oldOverrideValuesFile* is the file name and path to the old version of your **override-values.yaml** file for **oc-cn-ece-helm-chart**.

Failing Over Site 1 to Site 2

Site 2 has been upgraded at this stage, but it is not handling any usage requests. You now switch usage request processing from Site 1 to Site 2.

Figure 18-7 shows failing over Site 1 to Site 2.

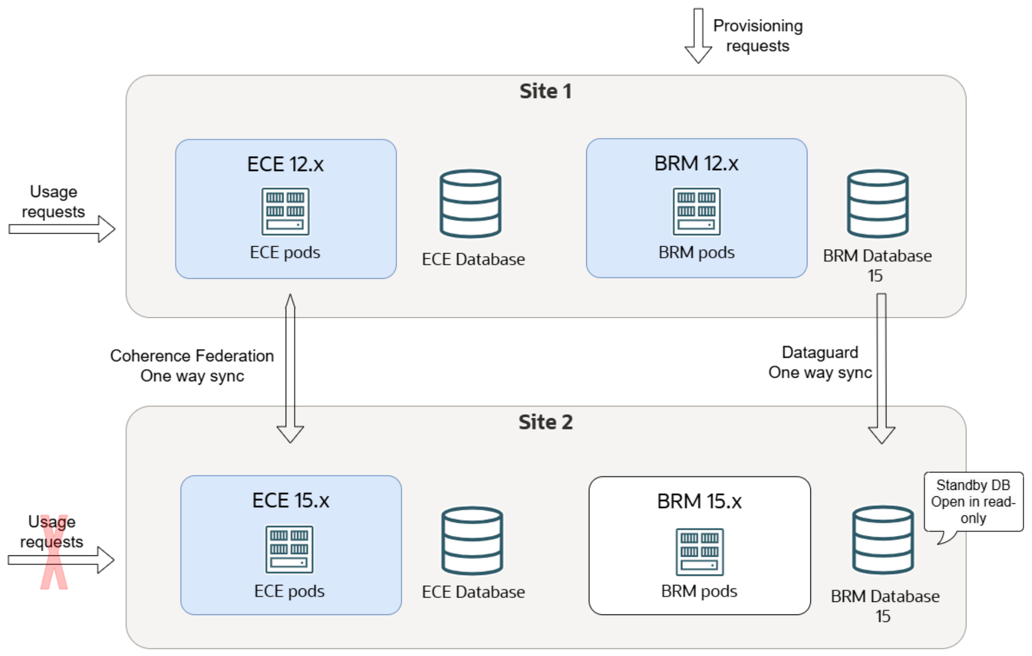
Figure 18-7 Failing Over Site 1 to Site 2



To fail over from Site 1 to Site 2:

1. Start the Coherence federation process from Site 1 to Site 2, as shown in Figure 18-8. This provisions the empty ECE Site 2 cache with the latest data from Site 1.

Figure 18-8 Coherence Federation Process from Site 1 to Site 2

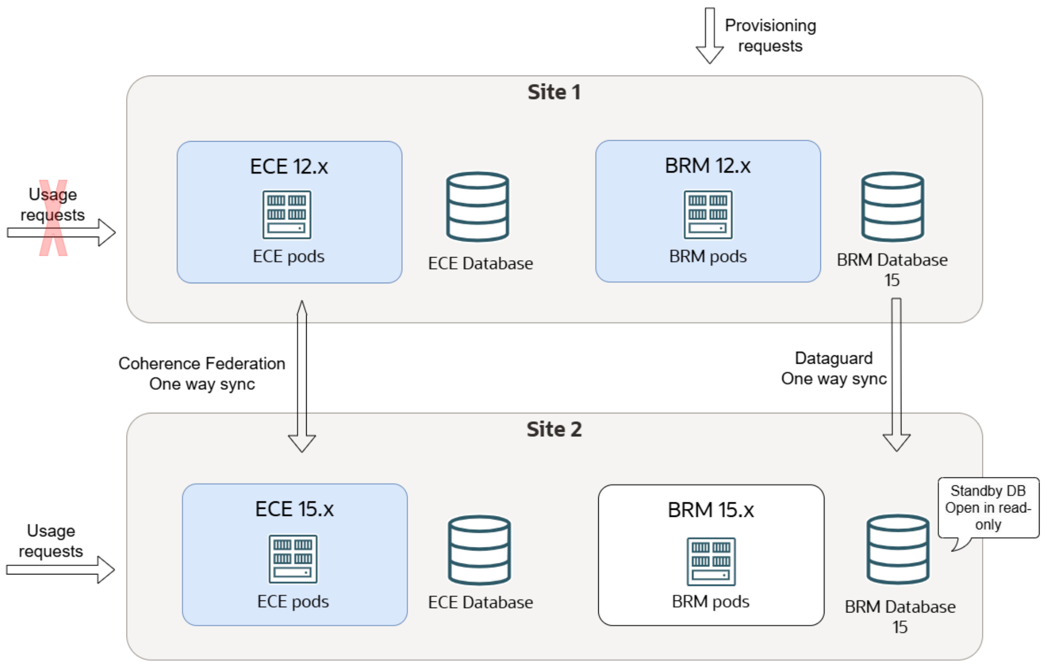


2. On Site 2, mark Site 1 inactive to ensure no preferred site routing occurs from Site 2 to Site 1.

> **Note:**
>
> Site 1 also has Site 2 marked as inactive.

3. Check that the federation process is up to date. After the federation process completes successfully:
   - The ECE pods in Site 2 transition to the **Running** state.
   - Site 2 transitions to the **Usage Processing** state and spawns Monitoring Agent pods.
4. Stop all usage requests to Site 1 and then redirect them to Site 2, as shown in Figure 18-9.

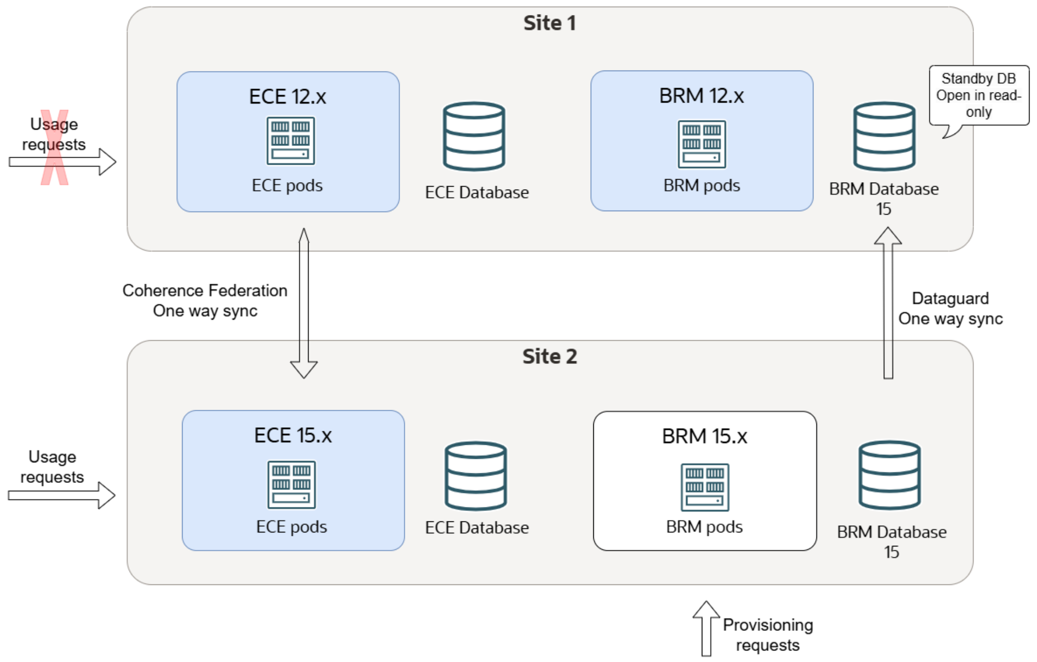
Figure 18-9 Usage Requests for Failing Over Site 1 to Site 2



5. Switch provisioning to BRM on Site 2, as shown in Figure 18-10. To do so:
   a. On BRM Site 2, remove any connections to EM Gateway on Site 1.
   b. On the client side, switch provisioning to BRM on Site 2.
   c. On ECE Site 1, stop the EM Gateway.

> **Note:**
>
> If latency between Site 1 and Site 2 is too high to give you acceptable performance for the provisioning flow, reverse the Oracle Data Guard roles and make the BRM database on Site 2 active. Be aware that this can cause a service interruption of a few minutes for provisioning.

Figure 18-10 Switched Provisioning to BRM on Site 2



6. Check that the federation process from Site 1 to Site 2 has completed.

   Now that all traffic to Site 1 has stopped, ensure all data from Site 1 has been synchronized with Site 2. The Coherence Federation Metrics should display **IDLE** instead of **YIELDING**.

7. Check that all rated events in the ECE Site 1 cache have been extracted by running the **query** utility:

   ```
   ./query.sh
   Coherence Command Line Tool
   CohQl> select value() from AggregateObjectUsage
   ```

   If successful, this command returns zero entries.

8. On Site 1, check that all Site 1 rated events have been extracted from the persistence database and are present in BRM by using SQL*Plus:

   ```
   sqlplus pin@databaseName
   Enter password: password

   SQL> select count(*) from ratedevent_site1Name
   ```

   where *databaseName* is the service name or database alias of the BRM database, and *password* is the password for the **pin** user.

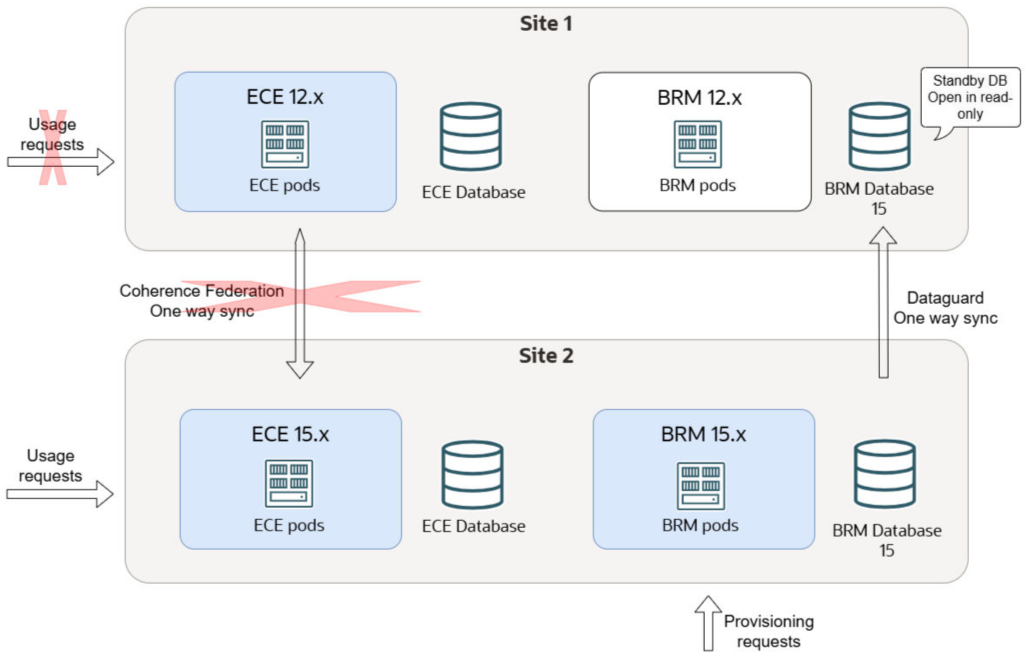
Note:

The Site 1 persistence database might contain some Site 2 events. After the Site 2 to Site 1 federation process is stopped, these events are not extracted or purged. However, they will be processed on Site 2. You can ignore these events because they get purged when you re-create the Site 1 persistence database later.

9. Stop the federation process from Site 1 to Site 2, as shown in Figure 18-11.

   Site 1 is isolated and ready for the upgrade.

Figure 18-11 Stopped Federation Process from Site 1 to Site 2



Uninstalling BRM and ECE from Site 1

Uninstall the old version of BRM and ECE from Site 1.

To do so, perform the following steps on Site 1:

1. Uninstall the old version of BRM cloud native:

```
helm uninstall BrmReleaseName --namespace BrmNameSpace
```

2. Uninstall the old version of ECE cloud native:

```
helm uninstall EceReleaseName --namespace BrmNameSpace
```

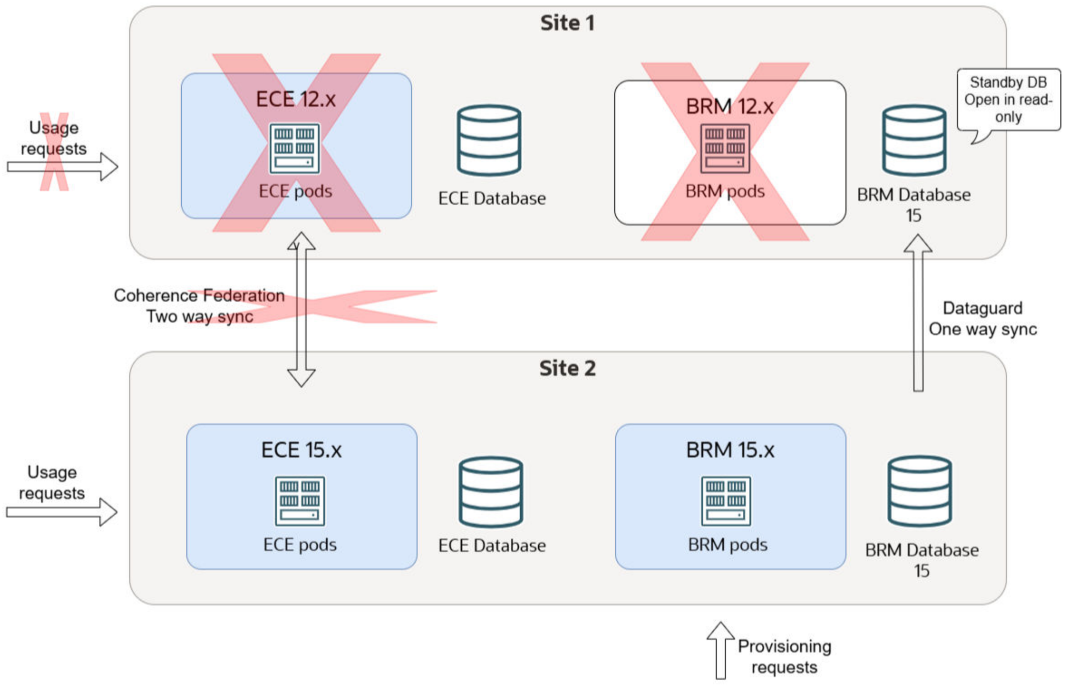
where:

- *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance.
- *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance.
- *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM Helm chart.

These commands delete all of the resources associated with the chart's last release and the release history.

Figure 18-12 shows uninstallation of BRM and ECE from Site 1.

Figure 18-12 Uninstallation of BRM and ECE from Site 1



Installing BRM Cloud Native on Site 1

To configure and deploy BRM 15.*x* cloud native on Site 1:

1. Download and extract the BRM 15.*x* cloud native Helm chart (**oc-cn-helm-chart**) and BRM Operator Job Helm Chart (**oc-cn-op-job-helm-chart**) from Oracle Software Delivery Cloud (https://edelivery.oracle.com).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files" for more information.
2. To reuse your old SSL KeyStore, copy the PDC KeyStore files from the old Helm chart to the 15.*x* **oc-cn-op-job-helm-chart/pdc/pdc_keystore/** directory.
3. Create an **override-values-15.yaml** file.

   You will use this file with the 15.*x* version of **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

4. In your **override-values-15.yaml** file, set the following keys:
   - In the BRM section:
     - **ocbrm.is_upgrade**: Set this to **true**.
     - **ocbrm.existing_rootkey_wallet**: Set this to **false**.
     - **ocbrm.db.***: Set the BRM database schema details to the same values as your old release.
   - In the PDC section:
     - **ocpdc.configEnv.pdcSchemaUserName**: Set this to the same value as your old release.
     - **ocpdc.configEnv.crossRefSchemaUserName**: Set this to the same value as your old release.
     - **ocpdc.configEnv.rcuPrefix**: Set this key to a new prefix to create a new RCU schema. The value must be different from the one used in Site 2.
     - **ocpdc.configEnv.transformation.upgrade**: Set this to **true**. (For upgrades to 15.0.0 only)
     - **ocpdc.secretValue.walletPassword**: Set this to the same value as your old release.
     - **ocpdc.configEnv.deployAndUpgradeSite2**: Set this to **false**. (For upgrades to 15.0.1 or later)
     - **ocpdc.configEnv.upgrade**: Set this to **true**. (For upgrades to 15.0.1 or later)
   - In the Business Operations Center section:
     - **ocboc.boc.configEnv.bocSchemaUserName**: Set this to the same value as your old release.
     - **ocboc.boc.configEnv.runUpgrade**: Set this to **true**.
     - **ocboc.boc.configEnv.rcuPrefix**: Set this to a new prefix for the RCU schema. The value must be different from the one used in Site 2.
     - **ocboc.boc.secretVal.***: Set the Business Operations Center passwords to the same values as your old release.
   - In the Billing Care section:
     - **ocbc.bc.configEnv.rcuPrefix**: Set this to a new prefix for the RCU schema. This value must be different from the one used in Site 2.
     - **ocbc.bc.secretVal.***: Set the passwords to the same values as your old release.
   - In the Billing Care REST API section:
     - **ocbc.bcws.configEnv.rcuPrefix**: Set this to a new prefix for the RCU schema. This value must be different from the one used in Site 2.
     - **ocbc.bcws.secretVal.***: Set the passwords to the same values as your old release.
5. Save and close the file.

6. Create WebLogic domains by running the 15.*x* version of **oc-cn-op-job-helm-chart** from the **helmcharts** directory:

```
helm install oldOpJobReleaseName oc-cn-op-job-helm-chart --namespace
oldBrmNameSpace --namespace oldOverrideValuesFile --values override-
values-15.yaml
```

   where:

   - *oldOpJobReleaseName* is the release name assigned to your old release of the **oc-cn-op-job-helm-chart** installation.
   - *oldBrmNameSpace* is the namespace for your old version of the BRM deployment.
   - *oldOverrideValuesFile* is the file name and path of your old version of the **override-values.yaml** file for **oc-cn-op-job-helm-chart**.

7. Install BRM cloud native services by running the 15.*x* version of **oc-cn-helm-chart** from the **helmcharts** directory:

```
helm install oldBrmReleaseName oc-cn-helm-chart --namespace
oldBrmNameSpace --values oldOverrideValuesFile --values override-
values-15.yaml
```

   where *oldBrmReleaseName* is the release name assigned to your old version of the **oc-cn-helm-chart** installation.

Dropping the ECE Persistence Database Schema from Site 1

To drop the old persistence database schema from Site 1:

1. Run the following command:

```
DROP USER schemaname CASCADE
```

2. Uninstall the old version of ECE Helm chart:

```
helm uninstall oldEceReleaseName
```

   oldEceReleaseName is the release name for the old version of **oc-cn-ece-helm-chart**.

3. Uninstall the old version of **ece-persistence-job** from your system by running this command:

```
kubectl delete job ece-persistence-job --namespace oldBrmNameSpace
```

   where *BrmNameSpace* is the namespace for the old version of the BRM Helm chart.

Note:

The ECE Helm chart re-creates the database schema when you install it later.

Installing ECE 15.*x* Cloud Native on Site 1

To install ECE 15.*x* cloud native on Site 1:

1. Download and extract the ECE 15.*x* cloud native (**oc-cn-ece-helm-chart**) from Oracle Software Delivery Cloud (https://edelivery.oracle.com).

   See "Downloading Packages for the BRM Cloud Native Helm Charts and Docker Files" for more information.
2. Create an **override-values-ece-15.yaml** file.
3. In the file, configure your ECE 15.*x* cloud native services by following the instructions in "Configuring ECE Services".
4. Deploy the ECE 15.*x* cloud native services by entering this command from the **helmcharts** directory:

   ```
   helm install oldEceReleaseName oc-cn-ece-helm-chart --namespace
   oldBrmNameSpace --values oldOverrideValuesFile --values override-values-
   ece-15.yaml
   ```

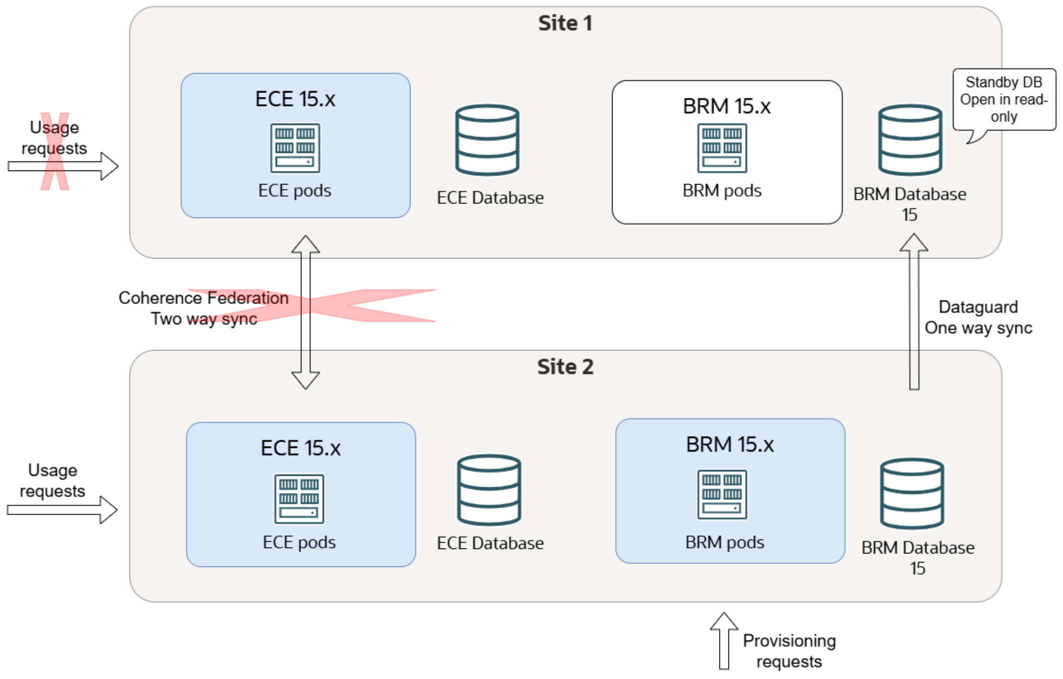
   where:

   - *oldEceReleaseName* is the release name for your old version of **oc-cn-ece-helm-chart**.
   - *oldBrmNameSpace* is the release name for your old version of **oc-cn-helm-chart** deployment.
   - *oldOverrideValuesFil*e is the file name and path to the old version of your **override-values.yaml** file for **oc-cn-ece-helm-chart**.

Site 1 is now upgraded, but it is idle.

Figure 18-13 shows installation of ECE on Site 1.

Figure 18-13   Installation of ECE Cloud Native on Site 1



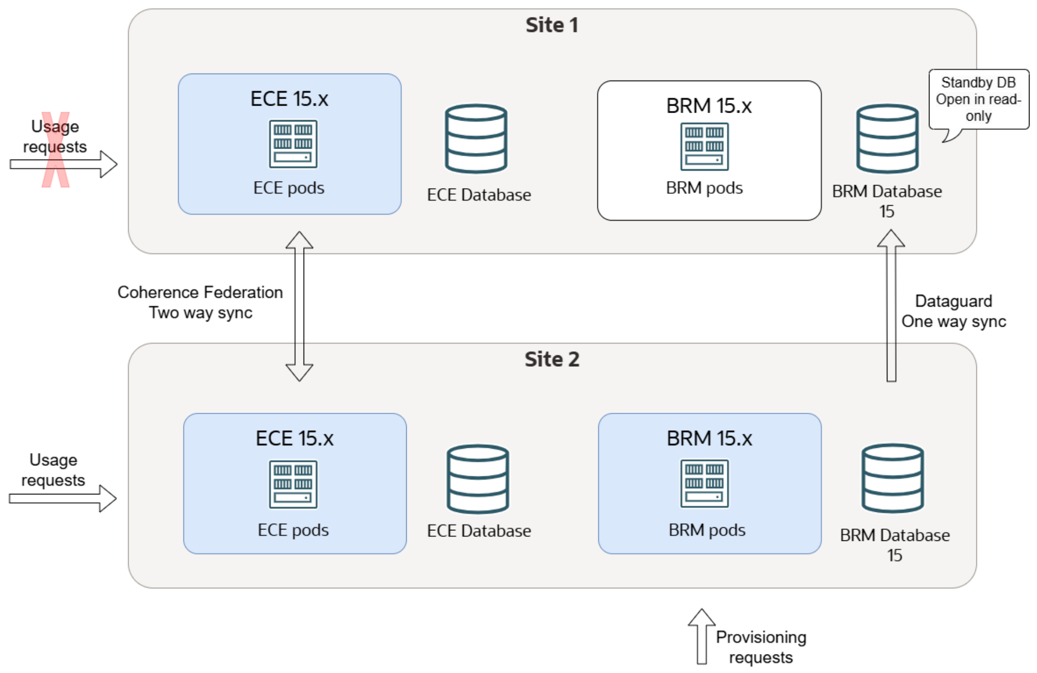
Federating ECE Cache Data Between Site 1 and Site 2

Now that Site 1 and Site 2 have been upgraded to release 15.*x*, you can restart the federation process between them.

To start the federation process of ECE cache data between Site 1 and Site 2:

1. Enable the two-way federation process between Site 1 and Site 2, as shown in Figure 18-14.

Figure 18-14   Federation of ECE Cache Data Between Sites



2. Check that the federation backlog is processed successfully.

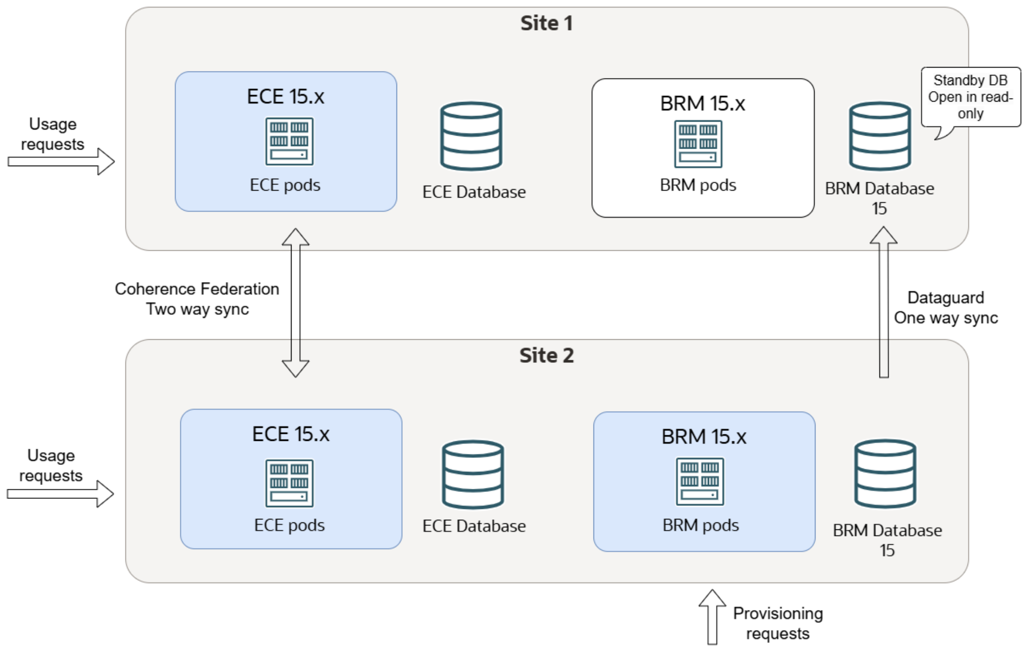
   After the federation process completes, ECE in Site 1 transitions to the **Usage Processing** state and spawns the Monitoring Agent pods.

3. In ECE Site 1, mark ECE in Site 2 as active. Likewise, in ECE Site 2, mark ECE in Site 1 as active.

   This enables usage rating requests to be forwarded to the subscriber's preferred site.

4. Enable usage requests to Site 2, as shown in Figure 18-15.

Figure 18-15   Enabled Usage Requests for Site 2



5. Enable failover connection to the EM Gateway. On both sides, ensure the CM has a failover EM connection to the EM Gateway on the other site.

19

Migrating from On-Premise BRM to BRM Cloud Native

Learn how to migrate from an on-premise release of Oracle Communications Billing and Revenue Management (BRM) 7.5.*x* or 12.*x* to a BRM 15.1 cloud native release.

Topics in this document:

- Migrating to BRM Cloud Native
- Migrating from PDC On Premises to PDC Cloud Native

Migrating to BRM Cloud Native

To migrate from an on-premises release to BRM cloud native:

1. Upgrade your BRM and PDC database schemas to release 15.1:
   - If you are upgrading from a 12.0 or 12.0 Patch Set 1 database schema, follow the instructions in "Installing BRM 12.0 Patch Sets" in *BRM Patch Set Installation Guide*.
   - If you are upgrading from a 12.0 Patch Set 2 or later database schema, follow the instructions in "Upgrading Your Database Schema".

   Note:

   - To use an existing BRM database schema with your cloud native deployment, follow the instructions in "Deploying BRM with an Existing Schema".
   - To use an existing Business Operations Center database schema, point to your existing Business Operations Center schema in the **override-values.yaml** file for **oc-cn-op-job-helm-chart**. See "Adding Business Operations Center Keys for oc-cn-op-job-helm-chart".

2. If you customized BRM or Billing Care, layer your customizations on top of the images provided with this release before deploying the images. For guidelines about customization, see "Customizing BRM Cloud Native Services".

Migrating from PDC On Premises to PDC Cloud Native

Before migrating your system, perform these steps:

- Ensure you have upgraded your PDC on-premises release to version 12.0 Patch Set 3 or later. Follow the instructions in "Upgrading Pricing Design Center Software".
- Back up your existing PDC and cross-reference database schemas.
- Download and load the PDC 15.1 cloud native images to the image repository or to the virtual machine (VM) where PDC is deployed.

- Download and extract the BRM 15.1 Helm charts.
- Verify that BRM has been upgraded to the same 15.1 cloud native version as PDC cloud native.

The following are the high-level steps for migrating PDC from an on-premises release to cloud native:

1. Stop your on-premises PDC domain.
2. In your **override-values.yaml** file for **oc-cn-op-job-helm-chart**, set the following keys to match the PDC and cross-reference database schema details for your existing on-premises release:

   ```
   ocpdc:
       configEnv:
           crossRefSchemaPDCTableSpace:
           crossRefSchemaTempTableSpace:
           crossRefSchemaUserName: UserName
           pdcSchemaPDCTableSpace:
           pdcSchemaTempTableSpace:
           pdcSchemaUserName: UserName
       secretValue:
           crossRefSchemaPassword: Password
           pdcSchemaPassword: Password
   ```

3. Set any other mandatory keys in your **override-values.yaml** file. For more information, refer to "Adding PDC Keys for oc-cn-op-job-helm-chart".
4. Deploy the PDC 15.1 domain to your cloud native environment:

   ```
   helm install OpJobReleaseName oc-cn-op-job-helm-chart --namespace
   BrmNameSpace --values OverrideValuesFile
   ```

   where:

   - *OpJobReleaseName* is the release name for **oc-cn-op-job-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.
   - *BrmNameSpace* is the namespace in which to create BRM Kubernetes objects for the BRM and Job Helm charts.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **oc-cn-op-job-helm-chart/values.yaml** file.

5. In your **override-values.yaml** file for **oc-cn-helm-chart**, set the **ocpdc.configEnv.upgrade** key to **true**.
6. Deploy the PDC 15.1 cloud native services to your cloud native environment, and ensure that the **pdc-domain** pod is in completed status:

   ```
   helm install BrmReleaseName oc-cn-helm-chart --namespace BrmNameSpace --
   values OverrideValuesFile
   ```

   where:

   - *BrmReleaseName* is the release name for **oc-cn-helm-chart** and is used to track this installation instance. It must be different from the one used for **oc-cn-op-job-helm-chart**.

- *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **oc-cn-helm-chart/values.yaml** file.

7. Ensure all files are upgraded successfully. To do so, check the status of the SQL upgrade log files in pdc-brm-pvc. If there is no failure message, all files upgraded successfully.

Part VI

Troubleshooting BRM Cloud Native Deployments

This part provides information about troubleshooting issues that may occur while deploying Oracle Communications Billing and Revenue Management (BRM) cloud native in your system. It contains the following chapters:

20 Troubleshooting Your BRM Cloud Native Deployment

Learn how to solve problems that may occur after the installation or upgrade of your Oracle Communications Billing and Revenue Management (BRM) cloud native system.

Topics in this document:

Problems with the Helm Installation

If a Helm installation encounters errors, such as an incorrect namespace, follow these steps to get back to a state where you can fix the issue and do a new installation.

Note:

For more information about Kubernetes commands, see "kubectl Cheat Sheet" in the Kubernetes documentation.

1. Check the state of the deployment:

   ```
   kubectl get pods --output wide --namespace NameSpace
   ```

   To see information about a specific pod:

   ```
   kubectl describe pod PodName --namespace NameSpace
   ```

2. Use the **helm rollback** command to go back to a previous revision of the chart, or use the **helm uninstall** command to uninstall the chart. See "Rolling Back A Release To A Previous Revision" in *BRM Cloud Native System Administrator's Guide*, or see "Helm Uninstall" in the Helm documentation.
3. If neither rolling back nor uninstalling the chart are successful, do the following to identify Kubernetes resources that did not install correctly and then delete them:
   - Check and delete all other stateful set components from the cluster:

     ```
     kubectl get sts
     ```

     If you identify a stateful set that you want to delete, scale the number of replicas:

     ```
     kubectl scale statefulsets StatefulSetName --replicas=n
     ```

where *StatefulSetName* is the name of a stateful set, and *n* is the number of replicas you are scaling to. For more information, see "Scale a StatefulSet" in the Kubernetes documentation.

Then, delete the stateful set:

```
kubectl delete StatefulSetName
```

You can run **kubectl get sts** again to verify the deletions.

- If you need to clean up Apache Kafka and Apache ZooKeeper, scale to 0 and then delete:

  ```
  kubectl scale sts/kafka_pod --replicas=0
  kubectl scale sts/zookeeper_pod --replicas=0
  kubectl get pods
  kubectl get sts
  kubectl delete sts kafka_pod zookeeper_pod
  ```

- If necessary, check any PVC, Secret, ConfigMap, or service that was created by the deployment. If the output from any of these commands shows something that you want to clean up, you can use **kubectl delete** to remove it.

  For example:

  ```
  kubectl get pvc --all-namespaces
  kubectl delete pvc PVCName

  kubectl get secrets --all-namespaces
  kubectl delete secret SecretName

  kubectl get configmap --all-namespaces
  kubectl delete configmap ConfigMapName

  kubectl get svc --all-namespaces
  kubectl delete svc SVC1 SVC2
  ```

Helm Installation Fails with Time-Out Error

After you deploy a Helm chart, you may receive the following error message indicating that the Helm chart installation failed:

```
Error: failed post-install: timed out waiting for the condition
```

This occurs because a post-installation job took longer than five minutes to complete.

To resolve the issue:

1. Purge your Helm release:

   ```
   helm delete BrmReleaseName --purge
   ```

   This removes and purges all resources associated with the last revision of the release.

2. Run the **Helm install** command again.

If that does not fix the problem, increase the amount of time Kubernetes waits for a command to complete by including the **--timeout** *duration* argument with the **helm install** command. For example, to set the timeout duration to 10 minutes, you would enter this command:

```
helm install BrmReleaseName oc-cn-helm-chart --namespace BrmNameSpace --timeout 10m --
values OverrideValuesFile
```

BRM Cloud Native Deployment Out of Memory Errors

After you deploy BRM cloud native, you may receive an error message similar to the following:

```
ERROR: cm_cache_heap_malloc: name="fm_bparams_cache" - out of memory, size
requested=2216,high val=960
cm_cache_flist: PIN_ERR_NO_MEM:requested=2216, used=121456, allocated=122880, chunk=30,
cache name="fm_bparams_cache"
```

To resolve the issue:

1. In your **oc-cn-helm-chart** directory, open your CM ConfigMap file (**configmap_pin_conf_cm.yaml**).
2. Add the following **fm_bparams_cache** entry to the file:

   ```
   - cm_cache fm_bparams_cache 40,245760,23
   ```

3. Save and close the file.
4. Run the **helm** upgrade command for **oc-cn-helm-chart**:

   ```
   helm upgrade BrmReleaseName oc-cn-helm-chart --values OverrideValuesFile --namespace
   BrmNameSpace
   ```

PDC Messages Stuck in Rating Engine Queues

Occasionally, PDC messages and changesets may become stuck in the rating engine queues.

To resolve the issue, delete both the RRE and BRE pods by running the following command:

```
kubectl --namespace BrmNameSpace delete pod PdcPodName
```

where:

- *BrmNameSpace* is the namespace in which the BRM Kubernetes objects reside.
- *PdcPodName* is the name of the pod.

Kubernetes automatically restarts the deleted pod, which restarts the transformation engine. Messages should start flowing again.

PDC Interceptor Pod is Started But Went to Error State

After you deploy PDC, the Interceptor pod may start but immediately transition to an error state.

This may occur because the RCU prefix is configured incorrectly. To find out if this is the case, run the following command:

```
kubectl describe domain DomainName --namespace NameSpace
```

If the issue is related to the RCU prefix, you will see something similar to the following:

```
WLSDPLY-12409: createDomain failed to create the domain: Failed to get FMW
infrastructure database defaults from the service table : Got exception when auto
configuring the schema component(s) with data obtained from shadow table:
Failed to build JDBC Connection object:
```

To resolve the issue:

1. Make sure that the RCU prefix is configured successfully as part of **oc-cn-op-job-helm-chart**. To do so, run the following command:

   ```
   kubectl get pod --namespace BrmNameSpace
   ```

   If it is configured correctly, **pdc-configure-rcu-***xxxxx* will show a Completed status.

2. Make sure that the RCU prefix and Password configured in the **override-values.yaml** file for **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart** matches, and that a valid host name, port, and service name have been configured in the **values.yaml** file.

If the values are not configured properly, do the following:

1. Uninstall PDC and then set the **ocpdc.isEnabled** key to **false** in your **override-values.yaml** file for **oc-cn-helm-chart**.
2. Run the **Helm upgrade** command for **oc-cn-helm-chart**.

   Wait until the PDC pods have stopped.
3. In the **override-values.yaml** file for **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**, configure the **ocpdc.configEnv.rcuPrefix** key and set the **ocpdc.isEnabled** key to **true**.
4. Run the **helm upgrade** command for **oc-cn-helm-chart** and **oc-cn-op-job-helm-chart**.

   Wait until the PDC pods are in Running status.

For more information about troubleshooting pod errors, see "Troubleshooting" in *Oracle WebLogic Kubernetes Operator Samples*.

eceTopology.conf Errors While Restarting Pods

While restarting the pricingupdater and brmgateway pods in your ECE cloud native deployment, you may receive an error message similar to the following:

```
ERROR MonitorFrameworkMessagesBundle-31300: Failed to initialize grid manager based on
topology file: eceTopology.conf property file:
ece.propertiesjava.lang.IllegalArgumentException: Nodes on a given host must be assigned
unique JMX ports
(check nodes: '[PricingUpdater node pricingupdater-6d575bf75b-r5q2t on Host
pricingupdater.ece-server.cluster,
PricingUpdater node pricingupdater-fbb9d7fb7-kqmxv on Host pricingupdater.ece-
server.cluster]')
```

This occurs because ECE cloud native has written invalid entries to the **eceTopology.conf** property file during the startup process.

To resolve the issue, do not restart the pricingupdater and brmgateway pods. Instead, scale down and then scale up those pods.

For example, to scale down and scale up the brmgateway pod:

1. Scale down the brmgateway pod to **0**:

   ```
   kubectl --namespace BrmNameSpace scale deploy brmgateway1 --replicas=0
   ```

   Wait for the brmgateway pod to stop.
2. Scale back up the brmgateway pod to **1**:

   ```
   kubectl --namespace BrmNameSpace scale deploy brmgateway1 --replicas=1
   ```