Oracle® Communications Order and Service Management

Cloud Native Deployment Guide for Oracle Application Integration Architecture Cartridges

Release 2.1.2

F43826-01

June 2021

Cloud Native Deployment for Oracle Application Integration Architecture Cartridges

Order-to-Activate 2.1.2.1.0 cartridges support deploying OSM instances in a cloud native environment. The integration pack generates all the required artifacts to be used by the OSM cloud native toolkit.

You perform the following tasks to use the Order-to-Activate cartridges for OSM cloud native deployments:

- Generate the OSM cloud native artifacts for the Order-to-Activate solution. See "Generating the OSM Cloud Native Artifacts for the Order-to-Activate Solution".
- Extend the OSM container image with the additional Order-to-Activate applications. See "Extending the OSM Docker Image with Additional Order-to-Activate Applications".
- Configure the specifications. See "Configuring the Specifications".
- Create the Order-to-Activate credentials and accounts. See "Creating Order-to-Activate Credentials and Accounts".
- Deploy the Order-to-Activate solution cartridge to the OSM cloud native instance's database. See *OSM Cloud Native Deployment Guide* for instructions about deploying cartridges to the OSM database.
- Bring up the OSM instance.

Generating the OSM Cloud Native Artifacts for the Order-to-Activate Solution

The Order-to-Activate integration pack generates all the required artifacts to be used by the OSM cloud native toolkit. You use the **config_ALL_CloudNative** configuration option in the **SolutionConfig.xml** Order-to-Activate configuration script to generate the artifacts.

Before generating the OSM cloud native artifacts, ensure that you have installed Design Studio and OSM plugins and that a workspace has already been created. For details about installing the Order-to-Activate components, see "Performing an

Interactive Installation of the Order-to-Activate Components" in *Order and Service Management Cartridge Guide for Oracle Application Integration Architecture*.

To generate the OSM cloud native artifacts:

1. Create a working folder and download the **OracleComms_OSM_O2A_CartridgesInstaller_2.1.2.1.0.xxx.zip** Order-to-Activate installer into the working folder.
2. Unzip the Order-to-Activate installer ZIP file within the working folder.
3. From Design Studio, import the *working_folder*/**OSM.PIP/ OracleComms_OSM_O2A_Install.zip** archive file as an existing project.
4. Import COM, SOM, or COM and SOM together with the solution cartridge into the workspace and upgrade the model. See "Performing an Interactive Installation of the Order-to-Activate Components" in *Order and Service Management Cartridge Guide for Oracle Application Integration Architecture*.
5. Add the **SolutionConfig.xml** file from the solution cartridge that you imported to Ant view and configure it to run as the same JRE as the Eclipse workspace.
6. Run the **config_All_CloudNative** Ant target.
7. When prompted, enter **d** for `development` or `p` for `production`.
8. Choose the topology and complex topology options as desired. The **config_All_CloudNative** target also runs the **config_Metadata_And_ModelVariable** target. Hence, there is no need to run it separately.
   After **config_All_CloudNative** is run successfully, the cloud native artifacts are generated. If you chose the CSO option, the artifacts are generated under **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native**.
9. Save the generated artifacts.
10. Build your solution. If you chose the CSO option, build **OracleComms_OSM_O2A_XXX_CSO_Solution**.

If you chose the CSO option, the par file is generated at: **OracleComms_OSM_O2A_XXX_CSO_Solution/cartridgeBin/ OracleComms_OSM_O2A_XXX_CSO_Solution.par**.

You use the OSM cloud native toolkit to deploy the par file to the OSM database directly.

For each solution, the generated cloud native artifacts include the following:

- **o2a_project.yaml**: The yaml fragment containing all JMS resources and SAF queues. Copy this fragment into your project specification file.
- **o2a_project-instance.yaml**: The yaml fragment containing all SAF connection configuration. Copy this fragment into your instance specification file.
- **o2a_users.yaml**: The yaml fragment of all cartridge users defined for the solution. Copy this fragment into your project specification file.
- **o2a_users_embedded_ldap.txt**: User information text file to be used by the **manage-cartridge-credentials.sh** OSM cloud native toolkit script to create

Kubernetes secrets for all cartridges. These secrets are added to WebLogic embedded LDAP when the instance is created.

- **o2a-wdt-app-archive.zip**: The application archive that contains the error handler for the solution. If the Order-to-Activate config option `development` or `cso` is selected, then emulators for AIA and the CSO solution are included as well.
- Dockerfile: The Docker build file to extend a base OSM image to include the **o2a-wdt-app-archive.zip** file.
- **create-o2a-base-image.sh**: The Linux shell script that uses the application archive and dockerfile to extend an OSM image for Order-to-Activate use.

Extending the OSM Docker Image with Additional Order-to-Activate Applications

To extend the base OSM Docker image:

1. Change the working directory to the directory where the cloud native artifacts are generated. If you chose CSO, change the working directory to **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native/application**.
2. Build the new Docker image containing the archive by running the **create-o2a-base-image.sh** script.

   The following sample shows the usage:

```
Usage:
create-o2a-base-image.sh <parameters>
 -t o2a-image-tag-name : mandatory, the image tag name to be
created.
 -i osm-base-image-name : mandatory, the base osm image to be
extended.
 -a o2a-app-archive : mandatory, the application archive that
contains O2A app artifacts.

For example....
$ ./create-o2a-base-image.sh \
-t o2a-domain:7.4.1 -i osm-base:7.4.1 -a o2a-wdt-app-archive.zip
```

Configuring the Specifications

In addition to the regular setup required for an instance defined by the OSM cloud native toolkit, you must perform the following additional steps:

1. For both production and development environments, merge the contents of your Order-to-Activate solution project yaml file into your *project_name*.**yaml** file. If you chose CSO, merge the contents of the **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native/project/o2a_project.yaml** file into your *project_name*.**yaml** file.
2. For production environments, if you chose product mode when running **config_ALL_CloudNative**, merge the contents of your Order-to-Activate solution

project file OR the **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native/project/o2a_project-instance.yaml** file into your *project_name-instance_name*.**yaml** file.

3. Edit the specification files as follows:

```
$ vi spec_path/project_name.yaml # update image and add credential
store as shown.

image: O2A_image_name_and_tag

#External Credentials Store, O2A cartridge user secret
externalCredStore:
 # Uncomment and specify Kubernetes secret suffix for external/peer
applications. The prefix project_name-instance_name will be derived
 # The suffix must be in the "osmcn-cred-mapName" format
 secrets:
   mapNames:
     - osm
     - uim
```

If you chose product mode when running **config_ALL_CloudNative**, modify your *project_name-instance_name*.**yaml** to add `customSecrets` and `safConnectionConfig` sections to set up the secrets and SAF end point URL for the SAF connection.

```
$ vi spec_path/project_name-instance_name.yaml
# add the Order-to-Activate credentials store as shown

#Customer Credentials Store, a secret used in SAF remote connect to
AIA, UIM and TOM
# Replace all ${DOMAIN_UID} with project_name-instance_name
instance:
  customSecrets:
    secretNames:
      - "${DOMAIN_UID}-saf-credentials-aia"
      - "${DOMAIN_UID}-saf-credentials-uim"
      - "${DOMAIN_UID}-saf-credentials-tom"


$ vi spec_path/project_name-instance_name.yaml
 # provide the AIA t3Url under O2A_SAFImportedDestinations,
 # UIM t3Url under O2A_UIM_SAFImportedDestinations and
 # TOM t3Url under O2A_TOM_SAFImportedDestinations.
 # Replace all ${DOMAIN_UID} with project_name-instance_name

# SAF connection configuration for O2A
safConnectionConfig:
 - name: O2A_SAFImportedDestinations
   t3Url: t3://t3_host:t3_port
   secretName: ${DOMAIN_UID}-saf-credentials-aia
 - name: O2A_UIM_SAFImportedDestinations
```

```
      t3Url: t3://t3_host:t3_port
      secretName: ${DOMAIN_UID}-saf-credentials-uim
    - name: O2A_TOM_SAFImportedDestinations
      t3Url: t3://t3_host:t3_port
      secretName: ${DOMAIN_UID}-saf-credentials-tom
```

Creating Order-to-Activate Credentials and Accounts

You create the following secrets to carry credential and account information for Order-to-Activate:

- Secrets used by SAF connection to AIA, UIM, and OSM TOM
- Secrets used by automation in OSM and Order-to-Activate to communicate to OSM and other systems

Creating Secrets for SAF Connection to AIA, UIM and OSM TOM

> **Note:**
>
> Perform this task for production environments only. When configuring the `config_ALL_CloudNative` option, if you chose `development`, the solution uses emulators and you do not need to create secrets for SAF connection.

The credentials used to obtain connection to other systems via SAF are stored as Kubernetes secrets.

To create the secrets for SAF remote context to AIA, UIM, and TOM, run the following command:

```
kubectl create secret generic project_name-instance_name-saf-
credentials-aia -n project_name --from-
literal=username='aia_weblogic_user_name' --from-
literal=password='aia_weblogic_user_password'

kubectl create secret generic project_name-instance_name-saf-
credentials-uim -n project_name --from-
literal=username='uim_weblogic_user_name' --from-
literal=password='uim_weblogic_user_password'

kubectl create secret generic project_name-instance_name-saf-
credentials-tom -n project_name --from-
literal=username='tom_weblogic_user_name' --from-
literal=password='tom_weblogic_user_password'
```

Creating Secrets for Order-to-Automate Automation Users

The Order-to-Activate user credentials are persisted to a cartridge credential secret in Kubernetes. This uses the `osm` map name and is available via the OSM automation

framework in the cartridge automation plugin code. In addition, the automation users must have accounts in embedded LDAP.

To create the secret and embedded user accounts:

1. Populate your *project*.**yaml** file with the list of `cartridgeUsers` from the **o2a_users.yaml** file. If you chose CSO, the list of cartridge users is located in the **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native/project/o2a_users.yaml** file.

   ```
   # cartridge users for o2a
   cartridgeUsers:
    - osm
    - osmoe
    - osmde
    - osmfallout
    - osmoelf
    - osmlfaop
    - osmlf
   ```

2. Copy the **o2a_users_embedded_ldap.txt** Order-to-Activate user file to **$OSM_CNTK/samples/credentials.** If you chose CSO, the user file is located in the **OracleComms_OSM_O2A_XXX_CSO_Solution/cloud-native/project** directory.
3. Run the **$OSM_CNTK/samples/credentials/manage-cartridge-credentials.sh** script to create the Kubernetes secret:

   ```
   $ chmod 777 $OSM_CNTK/samples/credentials/manage-cartridge-
   credentials.sh
   $ $OSM_CNTK/samples/credentials/manage-cartridge-credentials.sh -p
   project_name -i instance_name -c create -f $OSM_CNTK/samples/
   credentials/o2a_users_embedded_ldap.txt
   ```

Oracle® Communications Order and Service Management Cloud Native Deployment Guide for Oracle Application Integration Architecture Cartridges, Release 2.1.2
F43826-01