Oracle® Communications
Cloud Native Unified Data Repository Installation and Upgrade Guide

Release 1.6
F30716-02
May 2020

Oracle Communications Cloud Native Unified Data Repository Installation and Upgrade Guide, Release 1.6

F30716-02

Contents

What's New in This Guide

This section shares the list of new features introduced in every OCUDR release. For more release specific information, please refer to its release notes.

Release 1.6
The following new features are introduced in this release:

- Supports AM, SM and UEPolicy set as per 29.519 v16.2.0
- HTTPs/TLS support using Ingress API gateway
- CNC-Console integration on provisioning APIs: Refer to UDR Users Guide
- Provisioning Gateway integration for SLF provisioning: Refer to *UDR Provisioning Gateway Guide* for more details.
- Metrics for provisioned data

List of Tables

1 Introduction

This documents provides information for installing Cloud Native Unified Data Repository product.

Overview

The 5G Unified Data Repository (UDR) is one of the main key component of the 5G Service Based Architecture. UDR is a converged repository, which is used by 5G Network Functions to store the data.

Oracle 5G UDR is implemented as cloud native function and it offers a unified database for storing application, subscription, authentication, service authorization, policy data, session binding and Application state information. It exposes a HTTP2 based RESTful API for NF's and provisioning clients to access the stored data.

Oracle's 5G UDR:

- Leverages a common Oracle Communications Cloud Native Framework
- Is compliant to 3GPP Release 15 specification UDM
- Is compliant to 3GPP v29.519 v16.2 (backward compatible with v15.3.0, by configuration) specification for PCF
- Has tiered architecture providing separation between the connectivity, business logic and data layers
- Uses Oracle MySQL Cluster CGE backend database provides through DB Tier.
- Registers with NRF in the 5G network, so the other NFs in the network can discover UDR through NRF.

As per 3GPP, UDR supports following functionality:

- Storage and retrieval of subscription data by the UDM.
- Storage and retrieval of policy data by the PCF.
- Storage and retrieval of structured data for exposure.
- Storage and retrieval of SLF information, consumed by NRF.
- Application data (including Packet Flow Descriptions (PFDs) for application detection, AF request information for multiple UEs), by the NEF.
- Subscription and Notification feature.

Oracle's 5G UDR provides Unstructured Data Storage Function (UDSF) functionality. This functionality:

- Supports storage and retrieval of unstructured data by any 5G NF. The specifications of UDSF are presently not defined by 3GPP completely.
- This functionality is part of Oracle's 5G UDR solution.

Oracle's 5G UDR provides 5G SLF functionality. This functionality:

- Supports Nudr-groupid-map service as defined by 3GPP
- Complaint with 3GPP Release 16 for APIs to be consumed by 5G NRF
- Supports REST/JSON based provisioning APIs for SLF data

Architecture

The Cloud Native Unified Data Repository architecture has following three tiers:

Connectivity Tier

- Ingress API Gateway (Spring Cloud Gateway [SCG] based) is used as an API gateway that receives all requests and forwards them to the Nudr-drservice service of Business Tier.
- It load balances the traffic and provides required authentication.
- It provides TLS support.

Business Tier

- Provides the business logic of 5G Unified Data Repository.
- It has following three micro services:
  - **nudr-drservice:** The core service that handles flexible URI support, runtime schema validation and connects to Data Tier for DB operations. It provides SLF lookup functionality.
  - **nudr-nrf-client-service:** Handles registration, heartbeat, update and deregistration with Network Repository Function (NRF).
  - **nudr-notify-service:** Handles notification messages to Policy Control Function (PCF) and Unified Data Management (UDM) for data subscriptions.

Data Tier

- Uses Oracle MySQL NDB Cluster, CGE edition as backend database in the DB tier. This provides HA and geo-redundcancy capabilities.

References

Refer to the following documents for more information about 5G cloud native unified data repository.

- CNE Installation Guide
- PCF Installation Guide
- NRF Installation Guide
- Provisioning Gateway Guide
- Unified Data Repository User Guide

My Oracle Support

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

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

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

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

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

2

Installing Unified Data Repository

This section provides instructions on installing Unified Data Repository.

Planning Your Installation

Before installing UDR, perform the following pre-installation tasks:

- Checking the software requirements
- Checking the environment setup

Checking the Software Requirements
Before installing Unified Data Repository (UDR), install the following softwares on your system.

Software	Version
Kubernetes	v1.13.3
HELM	v2.12.3

Additional softwares that needs to be deployed as per the requirement of the services are:

Software	Version	Notes
elasticsearch	1.21.1	Needed for Logging Area
elastic-curator	1.2.1	Needed for Logging Area
elastic-exporter	1.1.2	Needed for Logging Area
logs	2.0.7	Needed for Logging Area
kibana	1.5.2	Needed for Logging Area
grafana	2.2.0	Needed for Metrics Area
prometheus	8.8.0	Needed for Metrics Area
prometheus-node-exporter	1.3.0	Needed for Metrics Area
metallb	0.8.4	Needed for External IP
metrics-server	2.4.0	Needed for Metric Server
tracer	0.8.3	Needed for Tracing Area

Note:

The above softwares are available in the **Oracle Communications Cloud Native Environment (OCCNE).** If you are deploying UDR in any other environment, then the above softwares must be installed before installing UDR.

To check the installed software items, execute the following command:

```
helm ls
```

Some systems may need to use helm command with **admin.conf** file as follows:

```
helm --kubeconfig admin.conf
```

> **Note:**
>
> Some of the above mentioned software(s) are updated frequently. Their later versions than those listed above should work with UDR 1.6. Some UDR features and services work differently depending on the software being used

.

Checking the Environment Setup
Before installing UDR, the system environment should have the following:

- **Access to OpenStack Environment:** User should have access to an existing OpenStack environment including the OpenStack Desktop. This environment is configured with appropriate resource flavors and network resources that allows its users to allocate resources to the virtual machines created via this procedure.
- **Availability of a pub key:** Users must have a pub key for logging into the Bootstrap Host. This key should be placed into the customer OpenStack Environment using **Import Key** tab on the **Launch Instance**→**Key Pair** dialog or via the **Compute**→**Access and Security**.
- **OCUDR Software:** User must install Kubernetes v1.13.3 and HELM v2.12.3. UDR consists of:
  - **Helm Charts** that reflect the OCUDR software version. It is a zipped tar file that you need to unzip.
  - **Docker images of the micro-services** that are shared as tar file. You need to untar it.

  > **Note:**
  >
  > For more details about OCUDR Software, see Checking the Software Requirements.

- **Create Database User/Group:** The Database administrator should create a user in the MYSQL DB using MySQL NDB cluster. UDR uses an NDB MySQL database to store the subscriber information. NDB MySQL database provides HA and geo-redundancy capabilities.
  The database administrator should also provide user with necessary permissions to access the tables in the NDB cluster. The steps to create a user and assign permissions are as follows:
  1. Login to the server where the ssh keys are stored and SQL nodes are accessible.
  2. Connect to the SQL nodes.
  3. Login to the Database as a root user.

4. Create a user on all sql nodes and assign it to a group having necessary permissions to access the tables on all sql nodes. Also, create a database on only one sql node.

```
CREATE USER '<username>'@'%' IDENTIFIED BY
'<password>';
DROP DATABASE if exists <db_name>;
CREATE DATABASE <db_name> CHARACTER SET utf8;
GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
TEMPORARY TABLES, DELETE, UPDATE,EXECUTE ON <db_name>.* TO
'<username>'@'%';
USE <db_name>;
```

> **Note:**
>
> You need this database name, username and password at the time of creating K8s secrets.

- **Network Access:** The Kubernetes cluster hosts must have network access to:
  - Local docker image repository where the Oracle Communications Unified Data Repository images are available.
    To check if the Kubernetes cluster hosts has network access to the local docker image repository, try to pull any image with tag name to check connectivity by executing:

    ```
    docker pull <docker-repo>/<image-name>:<image-tag>
    ```
  - Local helm repository where the Oracle Communications Unified Data Repository helm charts are available.
    To check if the Kubernetes cluster hosts has network access to the local helm repository, execute:

    ```
    helm repo update
    ```

> **Note:**
>
> Some of the systems may need to use helm command with `helm --kubeconfig admin.conf`

> **Note:**
>
> All the kubectl and helm commands (used in this document) must be executed on a system depending on the infrastructure of the deployment. It can be any client machine like virtual machine, server, local desktop and so on.

- **Laptop/Desktop Client Software:** A laptop/desktop where the user executes deployment commands should have:
  - Network access to the helm repository and docker image repository
  - Configuration of Helm repository on the client

- Network access to the Kubernetes cluster
- Necessary environment settings to run the `kubectl` commands. The environment should have privileges to create namespace in the Kubernetes cluster.
- Helm client installed with the **push** plugin. The environment should be configured so that the `'helm install'` command deploys the software in the Kubernetes cluster.

Note:

All the kubectl and helm commands (used in this document) must be executed on a system depending on the infrastructure of the deployment. It can be any client machine like virtual machine, server, local desktop and so on.

Installation Sequence

The installation sequence of UDR is as follows:

1. Installation Preparation
2. OCUDR Namespace Creation
3. Service Account, Role, and RoleBinding Creation
4. Creating Kubernetes Secrets for storing:
   - DBName, Username, Password and EncryptionKey
   - Private Keys and Certificate for IngressGateway
5. ocudr-custom-values.yaml File Configuration
6. UDR Deployment

Installation Preparation

This phase of installation includes downloading and loading the required files to the system.

1. Download the UDR package file from Oracle Software Delivery Cloud (OSDC). Execute the following command to download UDR package.
   `<nfname>-pkg-<marketing-release-number>.tgz`

   For example:`ocudr-pkg-1.6.0.0.0.tgz`
2. Untar the UDR Package File. Execute the following command to untar UDR Package File.
   `tar -xvf ocudr-pkg-1.6.0.0.0.tgz`

   This command results into `ocudr-pkg-1.6.0.0.0` directory. The directory consists of following:
   - **UDR Docker Images File:** `ocudr-images-1.6.0.tar`
   - **Helm File:** `ocudr-1.6.0.tgz`
   - **Readme txt File:** The `Readme.txt` contains cksum and md5sum of tarballs.

3. Verify the checksums of tarballs. Execute the following command:
   `Readme.txt`
4. Load the tarballs to docker images. Execute the following command:
   `# docker load --input /root/ocudr-images-1.6.0.tar`
5. Check if all the images are loaded. Execute the following command:
   `docker images | grep ocudr`
6. Tag the docker images to docker registry. Execute the following command:
   `docker tag <image-name>:<image-tag> <docker-repo>/<image-name>:<image-tag>`
7. Push the docker images to docker registry. Execute the following command:
   `docker push <docker-repo>/<image-name>:<image-tag>`

   Sample Tag and Push Commands:

   ```
   # docker tag ocudr/nudr_datarepository_service:1.6.0 <customer repo>/
   nudr_datarepository_service:1.6.0
   # docker push <customer repo>/nudr_datarepository_service:1.6.0
   # docker tag ocudr/nudr_nrf_client_service:1.6.0 <customer repo>/
   nudr_nrf_client_service:1.6.0
   # docker push <customer repo>/nudr_nrf_client_service:1.6.0
   # docker tag ocudr/nudr_notify_service:1.6.0 <customer repo>/
   nudr_notify_service:1.6.0
   # docker push <customer repo>/nudr_notify_service:1.6.0
   # docker tag ocudr/ocingress_gateway:1.6.2 <customer repo>/
   ocingress_gateway:1.6.2
   # docker push <customer repo>/ocingress_gateway:1.6.2
   # docker tag ocudr/configurationinit:1.1.1 <customer repo>/
   configurationinit:1.1.1
   # docker push <customer repo>/configurationinit:1.1.1
   # docker tag ocudr/configurationupdate:1.1.1 <customer repo>/
   configurationupdate:1.1.1
   # docker push <customer repo>/configurationupdate:1.1.1
   ```

8. Untar Helm Files. Execute the following command:
   `tar -xvzf ocudr-1.6.0.tgz`
9. Download the Unified Data Repository (UDR) Custom Template ZIP file from OHC. The steps are as follows:
   a. Go to the URL, docs.oracle.com
   b. Navigate to **Industries**->**Communications**->**Cloud Native Core**.
   c. Click the Unified Data Repository (UDR) Custom Template link to download the zip file.
   d. Unzip the template to get ocudr-custom-configTemplates-1.6.0.0.0 file that contains the following:
      - **UDR_Dashboard.json:** This file is used by grafana.
      - **ocudr-custom-values-1.6.0.yaml:** This file is used during installation.
      - **ProvGw_Dashboard.json**
      - **rollbackPCFschema_15_3.py**
      - **prov-gw5g-custom-values-1.6.0.yaml**

- **rollback.py**
- **upgrade.py**

Following are the OCUDR Images.

Table 2-1 OCUDR Images

Pod	Image
<helm_release_name>-nudr-drservice	ocudr/nudr_datarepository_service
<helm_release_name>-nudr-notify -service	ocudr/nudr_notify_service
<helm_release_name>-nudr-nrf-client-service	ocudr/nudr_nrf_client_service
<helm_release_name>-ingressgateway	ocudr/ocingress_gateway ocudr/configurationinit ocudr/configurationupdate

Note:

<helm_release_name>-nudr-notify-service is not required for SLF deployment. So, set its flag value as 'enabled - false' in the **values.yaml** file. For more details, see User Configurable Parameter.ocudr-custom-values.yaml File Configuration

OCUDR Namespace Creation

In this section, you will learn to verify the existence of a required namespace in the system. If a namespace does not exist, you must create it. The steps to verify and create a namespace are as follows:

1. Execute the following command to verify the existence of required namespace in system:
   `kubectl get namespace`
2. If the required namespace does not exist, then execute the following command to create a namespace:
   `kubectl create namespace <required namespace>`

   For example: `kubectl create namespace ocudr`

Note:

This is an optional step. In case required namespace already exists, proceed with next procedures.

Service Account, Role and RoleBinding Creation

In this section, you will learn to create a service account, role and rolebinding resources.

A sample command to create the resources is as follows:

```
kubectl -n <ocudr-namespace> create -f ocudr-sample-resource-template.yaml
```

A sample template to create the resources is as follows:

> **Note:**
>
> You need to update the <helm-release> and <namespace> values with its respective ocudr namespace and ocudr helm release name.

```
#
# Sample template start
#
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <helm-release>-serviceaccount
  namespace: <namespace>
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: <helm-release>-role
  namespace: <namespace>
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  verbs:
  - get
  - watch
  - list
---

apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: <helm-release>-rolebinding
  namespace: <namespace>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: <helm-release>-role
subjects:
- kind: ServiceAccount
  name:  <helm-release>-serviceaccount
  namespace: <namespace>
```

```
#
# Sample template end
#
```

Kubernetes Secret Creation - DBName, Username, Password and Encryption Key

In this section, you will learn to create a secret to store database name, username, password, and encryption key.

To create a Kubernetes secret:

1. Create a yaml file with dbname, dbusername, dbpassword, encryptionKey using the syntax given below:

   ```
   ocudr-secret.yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: ocudr-secrets
   type: Opaque
   data:
     dbname: dWRyZGI=
     dsusername: dWRydXNlcg==
     dspassword: dWRycGFzc3dk
     encryptionkey: TXkgc2VjcmV0IHBhc3NwaHJhc2U=
   ```

   The values of dbname, dsusername, dspassword, encryptionKey are base64 encoded. These are created by executing the following commands:

   ```
   echo -n "<db name>" | base64
   ```

   ```
   echo -n "<db username>" | base64
   ```

   ```
   echo -n "<db password>" | base64
   ```

   ```
   echo -n "<encryptionKey string>" | base64
   ```

   Note:

   You will create a secret using this yaml file.

2. Execute the following command to create a namespace where deployment is done.
   `kubectl create namespace <namespace>`

   Note:

   To create a secret, you need a namespace where deployment is done.

3. Execute the following command to create a secret:
   `kubectl create -f <secret File Name> -n <namespace>`

4. Execute the following command to verify a secret creation:
   ```
   kubectl describe secret <secret name> -n <namespace>
   ```

Kubernetes Secret Creation - Private Keys and Certificates for IngressGateway

In this section, you will learn to create a secret to store private keys and certificates for IngressGateway.

> **Note:**
>
> It is a user or operator discretion to create the private keys and certificates for IngressGateway and it is not in the scope of UDR. This section shares only samples to create them.

To create a secret to store private keys and certificate for IngressGateway:

1. Generate RSA private key by executing the following command:
   ```
   openssl req -x509 -nodes -sha256 -days 365 -newkey rsa:2048 -keyout
   rsa_private_key -out rsa_certificate.crt -config ssl.conf -passin
   pass:"keystorepasswd" -passout pass:"keystorepasswd"
   ```
2. Convert the private key to **.pem** format by executing the following command:
   ```
   openssl rsa -in rsa_private_key -outform PEM -out
   rsa_private_key_pkcs1.pem -passin pass:"keystorepasswd" -passout
   pass:"keystorepasswd"
   ```
3. Generate certificate using the private key by executing the following command:
   ```
   openssl req -new -key rsa_private_key -out apigatewayrsa.csr -config
   ssl.conf -passin pass:"keystorepasswd" -passout pass:"keystorepasswd"
   ```

   > **Note:**
   >
   > You can use **ssl.conf** to configure default entries along with storage area network (SAN) details for your certificate.

   A sample ssl.conf file is given below:

   ssl.conf
   ```
   #ssl.conf
   [ req ]
   default_bits = 4096
   distinguished_name = req_distinguished_name
   req_extensions = req_ext
   [ req_distinguished_name ]
   countryName = Country Name (2 letter code)
   countryName_default = IN
   stateOrProvinceName = State or Province Name (full name)
   stateOrProvinceName_default = Karnataka
   localityName = Locality Name (eg, city)
   localityName_default = Bangalore
   ```

```
organizationName = Organization Name (eg, company)
organizationName_default = Oracle
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_max = 64
commonName_default = localhost
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
IP = 127.0.0.1
DNS.1 = localhost
```

4. Create a root Certificate Authority (CA) by executing the following set of commands:
   ```
   openssl req -new -keyout cakey.pem -out careq.pem -config ssl.conf -
   passin pass:"keystorepasswd" -passout pass:"keystorepasswd"
   ```
   ```
   openssl x509 -signkey cakey.pem -req -days 3650 -in careq.pem -out
   caroot.cer -extensions v3_ca -passin pass:"keystorepasswd" echo 1234 >
   serial.txt
   ```
5. Sign the server certificate with root CA private key by executing the following command:
   ```
   openssl x509 -CA caroot.cer -CAkey cakey.pem -CAserial serial.txt -req
   -in apigatewayrsa.csr -out apigatewayrsa.cer -days 365 -extfile
   ssl.conf -extensions req_ext -passin pass:"keystorepasswd"
   ```
6. Generate ECDSA private key by executing the following set of commands:
   ```
   openssl ecparam -genkey -name prime256v1 -noout -out
   ecdsa_private_key.pem
   ```
   ```
   openssl pkcs8 -topk8 -in ecdsa_private_key.pem -inform pem -out
   ecdsa_private_key_pkcs8.pem -outform pem -nocrypt
   ```
7. Generate certificate using the private key by executing the following set of commands:
   ```
   openssl req -new -key ecdsa_private_key_pkcs8.pem -x509 -nodes -days
   365 -out ecdsa_certificate.crt -config ssl.conf
   ```
   ```
   openssl req -new -key ecdsa_private_key_pkcs8.pem -out
   apigatewayecdsa.csr -config ssl.conf -passin pass:"keystorepasswd" -
   passout pass:"keystorepasswd"
   ```
8. Sign the server certificate with root CA private key by executing the following command:
   ```
   openssl x509 -CA caroot.cer -CAkey cakey.pem -CAserial serial.txt -req
   -in apigatewayecdsa.csr -out apigatewayecdsa.cer -days 365 -extfile
   ssl.conf -extensions req_ext -passin pass:"keystorepasswd"
   ```
9. Create a key.txt file by entering any password.
   ```
   Example: echo "keystorepasswd" > key.txt
   ```
10. Create a trust.txt file by entering any password.
    ```
    Example: echo "truststorepasswd" > trust.txt
    ```
11. Create a Secret by executing the following set of commands:
    ```
    kubectl create ns NameSpace
    ```
    ```
    kubectl create secret generic ocudr-gateway-secret --from-
    file=apigatewayrsa.cer --from-file=caroot.cer --from-
    file=apigatewayecdsa.cer --from-file=rsa_private_key_pkcs1.pem --from-
    ```

```
file=ecdsa_private_key_pkcs8.pem --from-file=key.txt --from-
file=trust.txt -n <Namespace>
```

ocudr-custom-values.yaml File Configuration

In this section, you will learn to configure docker Registry path, DB connectivity service fqdn and port details and UDR details based on deployment.

UDR uses MySQL database to store the configuration and run time data. Before deploying the UDR in Kubernetes Cluster, update the following parameters in the **ocudr-custom-values-1.6.0.yaml** file:

Table 2-2 ocudr-custom-values-1.6.0.yaml Parameters

Section	Parameter	Services
Global	**mysql**	• **dbServiceName** : mysql-connectivity-service.occne-infra. • **port**: "<Port>".
	dockerRegistry: allows to configure docker Registry from where the images are pulled.	dockerRegistry: reg-1:5000
nrfclient	**host:**	• **baseurl**: "<To connect to Network Repository Function (NRF) for registration>". • **proxy**: "<Proxy setting if anyone connects to NRF>". Default value is NULL. • **capacityMultiplier**: "<Capacity Multiplier>". Default value is 500. • **supirange**: "<supi range for UDR>". Default value is [{\"start\": \"10000000000\", \"end\": \"20000000000\"}] • **priority**: "<priority>". Default value is 10. • **livenessProbeUrl**: "liveness probe url of nudr-drservice and nudr-notify-service". • **fqdn**: "FQDN of nudr-drservice for NRF to use while sending request. It is carried in registration request to NRF". • **gpsirange**: "<gpsi range for UDR>" • **plmnvalues**: "<plmn values that supports>"

Unified Data Repository Deployment

In this section, you will learn to deploy Unified Data Repository.

You can deploy UDR either with **HELM repository** or with **HELM tar**. To deploy UDR in Kubernetes cluster:

1. Use **ocudr-custom-values-1.6.0.yaml** file, which is modified in the ocudr-custom-values.yaml section. Execute the following command to deploy UDR:
   ```
   helm install <helm chart> [--version <OCUDR version>] --name <release>
   --namespace <k8s namespace> -f <ocudr-custom-values-1.6.0.yaml>
   ```
   In the above command:
   - *<helm chart>* - is the name of the chart, which is of the form <helm repo>/ocudr.
   - *<OCUDR version>* - is the software version (helm chart version) of the OCUDR. This is optional. If omitted, the default is **latest** version available in helm repository.
   - *<release>* - is a name of user's choice to identify the helm deployment. From 1.6.0 release onwards, all pod names, service name, deployment name are prepended by this release name.
   - *<k8s namespace>* - is a name of user's choice to identify the kubernetes namespace of the Unified Data Repository. All the Unified Data Repository micro services are deployed in this kubernetes namespace.
   - *<ocudr-custom-values-1.6.0.yaml>* - is the customized **ocudr-custom-values-1.6.0.yaml** file. The **ocudr-custom-values-1.6.0.yaml** file is a part of customer documentation. Users needs to download the file and modify it as per the user site.

   > **Note:**
   >
   > If helm3 is used, execute the following command for installation:
   > ```
   > helm install -name <release> --namespace <k8s namespace> -f
   > <ocudr-custom-values-1.6.0.yaml> <helm chart> [--version
   > <OCUDR version>]
   > ```

2. (Optional) Customize the Unified Data Repository by overriding the default values of various configurable parameters. See Customizing Unified Data Repository

Verifying UDR Deployment
After deploying UDR, you need to verify whether all the services and pods are up and running.

3 Customizing and Configuring Unified Data Repository

This section provides information on customizing and configuring Unified Data Repository.

Customizing Unified Data Repository

You can customize the Unified Data Repository deployment by overriding the default values of various configurable parameters.

In the ocudr-custom-values.yaml File Configuration section, MySQL host is customized.

The **ocudr-custom-values.yaml** file can be prepared by hand to customize the parameters.

Following is an example of Unified Data Repository customization file.

Note:

All the configurable parameters are mentioned in the Configuring User Parameters

```
OCUDR Customization File Collapse source
# Copyright 2019 (C), Oracle and/or its affiliates. All rights reserved.

global:
  dockerRegistry: udr-dev-bastion-1:5000
  mysql:
    dbServiceName: "mysql-connectivity-service.occne-infra"
#This is a read only parameter. Use the default value.
    port: "3306"
  jaeger:
    enabled: false
    host: "occne-tracer-jaeger-collector.occne-infra"
    port: 14268
  hikari:
    poolsize: "25"
  dbenc:
    shavalue: 256
  serviceAccountName:
  prefix:
    container:
    configmap:
    hpa:
```

```
nudr-drservice:
#  nameOverride: "nudr-drservice"
  image:
    name: ocudr/nudr_datarepository_service
    tag: 1.6.0
    pullPolicy: IfNotPresent

  service:
    http2enabled: "true"
    type: ClusterIP
    port:
      http: 5001
      https: 5002
      management: 9000

  notify:
    port:
      http: 5001
      https: 5002

  deployment:
    replicaCount: 2

  logging:
    level:
      root: "WARN"

  subscriber:
    autocreate: "true"

  validate:
    smdata: "false"

  resources:
    limits:
      cpu: 3
      memory: 4Gi
    requests:
      cpu: 3
      memory: 4Gi
    target:
      averageCpuUtil: 80

  minReplicas: 2
  maxReplicas: 4

nudr-notify-service:
#  nameOverride: "nudr-notify-service"
  enabled: true
  image:
    name: ocudr/nudr_notify_service
    tag: 1.6.0
    pullPolicy: IfNotPresent
  service:
    http2enabled: "true"
    type: ClusterIP
    port:
      http: 5001
      https: 5002
      management: 9000

  deployment:
    replicaCount: 2

  notification:
    retrycount: "3"
    retryinterval: "5"
    retryerrorcodes: "400,429,500,503"

  logging:
    level:
      root: "WARN"

  resources:
    limits:
      cpu: 3
      memory: 4Gi
    requests:
      cpu: 3
      memory: 4Gi
    target:
      averageCpuUtil: 80

  minReplicas: 2
  maxReplicas: 4

nudr-nrf-client-service:
#  nameOverride: "nudr-nrf-client-service"
  enabled: true
  host:
  baseurl: "http://ocnrf-ingressgateway.mynrf.svc.cluster.local/nnrf-
nfm/v1/nf-instances"
  proxy:
  ssl: "false"
  logging:
    level:
      root: "WARN"
  image:
    name: ocudr/nudr_nrf_client_service
    tag: 1.6.0
    pullPolicy: IfNotPresent
  heartBeatTimer: "90"
  groupId: "udr-1"
  capacityMultiplier: "500"
  supirange: "[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]"
  priority: "10"
  masterIp: "10.0.0.0"
  gpsirange: "[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]"
  endpointLabelSelector : "ocudr-ingressgateway"
  plmnvalues: "[{\"mnc\": \"14\", \"mcc\": \"310\"}]"
  scheme: "http"

 # The below 2 configuration will change based on site k8s name resolution
settings,
Also note the changes with namespace used for udr installation
  livenessProbeUrl: "http://nudr-notify-service.myudr.svc.cluster.
local:9000/actuator/health,http://nudr-drservice.myudr.svc.cluster.
local:9000/actuator/health"
  fqdn: "ocudr-ingressgateway.myudr.svc.cluster.local"

  resources:
    limits:
      cpu: 1
      memory: 2Gi
    requests:
      cpu: 1
      memory: 2Gi

ingressgateway:
 global:
   # Docker registry name
   # dockerRegistry: reg-1:5000

   # Specify type of service - Possible values are :- ClusterIP, NodePort,
LoadBalancer
and ExternalName
   type: LoadBalancer

   # Enable or disable IP Address allocation from Metallb Pool
   metalLbIpAllocationEnabled: true

   # Address Pool Annotation for Metallb
   metalLbIpAllocationAnnotation: "metallb.universe.tf/address-pool:
signaling"

   # If Static node port needs to be set, then set staticNodePortEnabled
flag to true
and provide value for staticNodePort
   #   # Else random node port will be assigned by K8
   staticNodePortEnabled: false
   staticHttpNodePort: 30075
   staticHttpsNodePort: 30043

 image:
   # image name
   name: ocudr/ocingress_gateway
   # tag name of image
   tag: 1.6.2
   # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
   pullPolicy: Always

 initContainersImage:
   # inint Containers image name
   name: ocudr/configurationinit
   # tag name of init Container image
   tag: 1.1.1
   # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
   pullPolicy: Always

 updateContainersImage:
   # update Containers image name
   name: ocudr/configurationupdate
   # tag name of update Container image
   tag: 1.1.1
   # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
   pullPolicy: Always

 service:
   ssl:
     tlsVersion: TLSv1.2

     privateKey:
       k8SecretName: ocudr-gateway-secret
       k8NameSpace: ocudr
       rsa:
         fileName: rsa_private_key_pkcs1.pem
       ecdsa:
         fileName: ecdsa_private_key_pkcs8.pem

     certificate:
       k8SecretName: ocudr-gateway-secret
       k8NameSpace: ocudr
       rsa:
         fileName: apigatewayrsa.cer
       ecdsa:
         fileName: apigatewayecdsa.cer

     caBundle:
       k8SecretName: ocudr-gateway-secret
       k8NameSpace: ocudr
       fileName: caroot.cer

     keyStorePassword:
       k8SecretName: ocudr-gateway-secret
       k8NameSpace: ocudr
       fileName: key.txt

     trustStorePassword:
       k8SecretName: ocudr-gateway-secret
       k8NameSpace: ocudr
       fileName: trust.txt

     initialAlgorithm: RSA256

 # Resource details
 resources:
   limits:
     cpu: 3
     memory: 4Gi
   requests:
     cpu: 3
     memory: 4Gi
   target:
     averageCpuUtil: 80

 log:
   level:
     root: WARN
     ingress: INFO
     oauth: INFO

 # enable jaeger tracing
 jaegerTracingEnabled: false

 openTracing :
   jaeger:
     udpSender:
       # udpsender host
       host: "occne-tracer-jaeger-query.occne-infra"
       # udpsender port
       port: 6831
     probabilisticSampler: 0.5

 # Number of Pods must always be available, even during a disruption.
 minAvailable: 2
 # Min replicas to scale to maintain an average CPU utilization
 minReplicas: 2
 # Max replicas to scale to maintain an average CPU utilization
 maxReplicas: 5

 # label to override name of api-gateway micro-service name
 #fullnameOverride: ocudr-endpoint

 # To Initialize SSL related infrastructure in init/update container
 initssl: false

 # Cipher suites to be enabled on server side
 ciphersuites:
   - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
   - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
   - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
   - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
   - TLS_DHE_RSA_WITH_AES_256_CCM
   - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
   - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

 #OAUTH CONFIGURATION
 oauthValidatorEnabled: false
 nfType: SMF
 nfInstanceId: 6faf1bbc-6e4a-4454-a507-a14ef8e1bc11
 producerScope: nsmf-pdusession,nsmf-event-exposure
 allowedClockSkewSeconds: 0
 nrfPublicKeyKubeSecret: nrfpublickeysecret
```

```
nrfPublicKeyKubeNamespace: ingress
validationType: strict
producerPlmnMNC: 123
producerPlmnMCC: 346

#Server Configuration for http and https support
#Server side http support
enableIncomingHttp: true
#Server side https support
enableIncomingHttps: false
#Client side https support
enableOutgoingHttps: false

maxRequestsQueuedPerDestination: 5000
maxConnectionsPerIp: 10


#Service Mesh (Istio) to take care of load-balancing
serviceMeshCheck: false
# configuring routes
routesConfig:
- id: traffic_mapping_http
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr/**
- id: traffic_mapping_http_prov
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr-prov/**
- id: traffic_mapping_http_mgmt
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr-mgm/**
- id: traffic_mapping_http_udsf
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudsf-dr/**
- id: traffic_mapping_http_group
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-group-id-map/**
- id: traffic_mapping_http_group_prov
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-group-id-map-prov/**
```

Configuring User Parameters

The UDR micro services have configuration options. The user should be able to configure them via deployment values.yaml.

Note:

The default value of some of the settings may change.

Note:

- **NAME**: is the release name used in helm install command
- **NAMESPACE**: is the namespace used in helm install command
- **K8S_DOMAIN**: is the default kubernetes domain (svc.cluster.local)

Default Helm Release Name:- ocudr

Following table provides the parameters for **global configurations**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
dockerRegistry	Docker registry from where the images will be pulled	reg-1:5000	Not applicable	
mysql.dbServiceName	DB service to connect	mysql-connectivity-service.occne-infra	Not applicable	This is a CNE service used for db connection. Default name used on CNE is the same as configured.
mysql.port	Port of DBService Connection	3306	Not applicable	
jaeger.service.name	Jaegar Service Name installed in CNE	occne-tracer-jaeger-collector.occne-infra	Not applicable	
jaeger.service.port	Jaegar Service Port installed in CNE	9411	Not applicable	
hikari.poolsize	Connection pool size	25	Not Applicable	The hikari pool connection size to be created at start up
dbenc.shavalue	Encryption Key size	256	256 or 512	
serviceAccountName	Service account name	null	Not Applicable	The serviceaccount, role and rolebindings required for deployment should be done prior installation. Use the created serviceaccountname here.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
prefix.container	Container configurable prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for container names on different pods of UDR deployment. If not configured, release name will be used as preifx.
prefix.configmap	Configmap configurabe prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for configmap names. If not configured, release name will be used as preifx.
prefix.hpa	HPA configurable prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for HPA names. If not configured, release name will be used as preifx.

Following table provides the parameters for **nudr-drservice micro service**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
image.name	Docker Image name	ocudr/ nudr_datarepository_service	Not applicable	
image.tag	Tag of Image	1.6.0	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
subscriber.autocreate	Flag to enable auto creation of subscriber	true	true/false	This flag will enable auto creation of subscriber when creating data for a non existent subscriber.
validate.smdata	Flag to enable correlation feature for smdata	false	true/false	This flag will control the correlation feature for smdata. This flag must be false if using v16.2.0 for PCF data.
logging.level.root	Log Level	WARN	Possible Values - WARN INFO DEBUG	Log level of the nudr-drservice pod
deployment.replicaCount	Replicas of nudr-drservice pod	2	Not applicable	Number of nudr-drservice pods to be maintained by replica set created with deployment
minReplicas	Minimum Replicas	2	Not applicable	Minimum number of pods
maxReplicas	Maximum Replicas	4	Not applicable	Maximum number of pods
service.http2enabled	Enabled HTTP2 support flag for rest server	true	true/false	Enable/Disable HTTP2 support for rest server
service.type	UDR service type	ClusterIP	Possbile Values- ClusterIP NodePort LoadBalancer	The kubernetes service type for exposing UDR deployment Note: Suggested to be set as ClusterIP (default value) always
service.port.http	HTTP port	5001	Not applicable	The http port to be used in nudr-drservice service
service.port.https	HTTPS port	5002	Not applicable	The https port to be used for nudr-drservice service
service.port.management	Management port	9000	Not applicable	The actuator management port to be used for nudr-drservice service

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.cpu	Cpu Allotment for nudr-drservice pod	3	Not applicable	The cpu to be allocated for nudr-drservice pod during deployment
resources.requests.memory	Memory allotment for nudr-drservice pod	4Gi	Not applicable	The memory to be allocated for nudr-drservice pod during deployment
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
notify.port.http	HTTP port on which notify service is running	5001	Not applicable	
notify.port.https	HTTPS port on which notify service is running	5002	Not applicable	

Following table provides the parameters for **nudr-notify-service micro service**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	flag for enabling or disabling nudr-notify-service	true	true or false	For SLF deployment, this micro service must be disabled.
image.name	Docker Image name	ocudr/nudr_notify_service	Not applicable	
image.tag	Tag of Image	1.6.0	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
notification.retrycount	Number of notifications to be attempted	3	Range: 1 - 10	Number of notification attempts to be done in case of notification failures. Whether retry should be done will be based on notification.retryerrorcodes configuration.
notification.retryinterval		5	Range: 1 - 60 Unit: Seconds	The retry interval for notifications in case of failure. Unit is in seconds. Whether retry should be done will be based on notification.retryerrorcodes configuration.
notification.retryerrorcodes	Notification failures eligible for retry	"400,429,500,503"	Valid HTTP status codes comma seperated	Comma separated error code should be given. These error codes will be eligible for retry notifications in case of failures.
logging.level.root	Log Level	WARN	Possible Values - WARN INFO DEBUG	Log level of the notify service pod
deployment.replicaCount	Replicas of nudr-notify-service pod	2	Not applicable	Number of nudr-notify-service pods to be maintained by replica set created with deployment
minReplicas	Minimum Replicas	2	Not applicable	Minimum number of pods
maxReplicas	Maximum Replicas	4	Not applicable	Maximum number of pods
service.http2enabled	Enabled HTTP2 support flag	true	true/false	This is a read only parameter. Do not change this value

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
service.type	UDR service type	ClusterIP	Possbile Values- ClusterIP NodePort LoadBalancer	The kubernetes service type for exposing UDR deployment Note: Suggested to be set as ClusterIP (default value) always
service.port.http	HTTP port	5001	Not applicable	The http port to be used in notify service to receive signals from nudr-notify-service pod.
service.port.https	HTTPS port	5002	Not applicable	The https port to be used in notify service to receive signals from nudr-notify-service pod.
service.port.management	Management port	9000	Not applicable	The actuator management port to be used for notify service.
resources.requests.cpu	Cpu Allotment for nudr-notify-service pod	3	Not applicable	The cpu to be allocated for notify service pod during deployment
resources.requests.memory	Memory allotment for nudr-notify-service pod	4Gi	Not applicable	The memory to be allocated for nudr-notify-service pod during deployment
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA

Following table provides the parameters for **nudr-nrf-client-service micro service**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	flag for enabling or disabling nudr-nrf-client-service	true	true/false	
host.baseurl	NRF url for registration	http://ocnrf-ingressgateway.mynrf.svc.cluster.local/nnrf-nfm/v1/nf-instances	Not applicable	Url used for udr to connect and register with NRF
host.proxy	Proxy Setting	NULL	nrfClient.host	Proxy setting if required to connect to NRF
ssl	SSL flag	false	true/false	SSL flag to enable SSL with udr nrf client pod
logging.level.root	Log Level	WARN	Possible Values - WARN INFO DEBUG	Log level of the UDR nrf client pod
image.name	Docker Image name	ocudr/nudr_nrf_client_service	Not applicable	
image.tag	Tag of Image	1.6.0	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
heartBeatTimer	Heart beat timer	90	Unit: Seconds	
groupId	Group ID of UDR	udr-1	Not applicable	
capacityMultiplier	Capacity of UDR	500	Not applicable	Capacity multiplier of UDR based on number of UDR pods running
supirange	Supi Range supported with UDR	[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]	Valid start and end supi range	
priority	Priority	10	Priority to be sent in registration request	Priority to be sent in registration request

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbeUrl	Liveness probe URL for nudr-drservice/api-gateway	http://nudr-notify-service.myudr.svc.cluster.local:9000/actuator/health http://nudr-drservice.myudr.svc.cluster.local:9000/actuator/health	Not Applicable	URL used by nrf-client-service to check liveness probe of nudr-drservice, nudr-notify-service and ocudr-ambassador pods. **Note:** Be cautious in updating this value. Should consider helm release name, namespace used for udr deployment and name resolution setting in k8s.
fqdn	UDR FQDN	ocudr-ingressgateway.myudr.svc.cluster.local	Not Applicable	FQDN to used for registering in NRF for other NFs to connect to UDR. Note: Be cautious in updating this value. Should consider helm release name, namespace used for udr deployment and name resolution setting in k8s.
gpsirange	Gpsi Range supported with UDR	[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]	Valid start and end gpsi range	
endpointLabelSelector	Pod name of ingress gateway	ocudr-ingressgateway	This should be changed based on the name space that you created.	End Point Label Selector is used to get the port number of the running ingress gateway pod that is deployed.
masterIp	Master IP of which we deployed	10.0.0.0	This should be changed with the master ip which we deployed	Master IP is used to send the ipv4 address to the nrf while registration.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
plmnvalues	Plmn values range that it supports	[{\"mnc\": \"14\", \"mcc\": \"310\"}]	This values can be changed that the range it supports	Plmn values are sent to nrf during regisration from UDR.
scheme	scheme in which udr supports	http	This can be changed to https.	scheme which we send to NRF during registration
resources.requests.cpu	Cpu Allotment for nudr-notify-service pod	1	Not applicable	The cpu to be allocated for nrf client service pod during deployment
resources.requests.memory	Memory allotment for nudr-notify-service pod	2Gi	Not applicable	The memory to be allocated for nrf client service pod during deployment
resources.limits.cpu	Cpu allotment limitation	1	Not applicable	
resources.limits.memory	Memory allotment limitation	2Gi	Not applicable	

Following table provides parameters for **ocudr-ingressgateway micro service (API Gateway)**

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
type	ocudr-ingressgateway service type	LoadBalancer	Possbile Values- ClusterIP NodePort LoadBalancer	
metalLbIpAllocationEnabled	Enable or disable Address Pool for Metallb	true	true/false	
metalLbIpAllocationAnnotation	Address Pool for Metallb	metallb.universe.tf/address-pool: signaling	Not applicable	
staticNodePortEnabled	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort	false	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
staticHttpNodePort	static http node port value need to be provided	30075	can be changed based of user requirement.	
staticHttpsNodePort	static https node port value need to be provided	30043	can be changed based of user requirement.	
image.name	Docker image name	ocudr/ ocingress_gateway	Not applicable	
image.tag	Image version tag	1.6.2	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
initContainersImage.name	Docker Image name	ocudr/ configurationinit	Not applicable	
initContainersImage.tag	Image version tag	1.1.1	Not applicable	
initContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
updateContainersImage.name	Docker Image name	ocudr/ configurationupdate	Not applicable	
updateContainersImage.tag	Image version tag	1.1.1	Not applicable	
updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
service.ssl.privateKey.k8SecretName	name of the secret which stores keys and certificates	ocudr-gateway-secret	Not applicable	
service.ssl.privateKey.k8NameSpace	namespace in which secret is created	ocudr	Not applicable	
service.ssl.privateKey.rsa.fileName	rsa private key stored in the secret	rsa_private_key_pkcs1.pem	Not applicable	
service.ssl.privateKey.ecdsa.fileName	ecdsa private key stored in the secret	ecdsa_private_key_pkcs8.pem	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
service.ssl.certificate.k8SecretName	name of the secret which stores keys and certificates	ocudr-gateway-secret	Not applicable	
service.ssl.certificate.k8NameSpace	namespace in which secret is created	ocudr	Not applicable	
service.ssl.certificate.rsa.fileName	rsa certificate stored in the secret	apigatewayrsa.cer	Not applicable	
service.ssl.certificate.ecdsa.fileName	ecdsa certificate stored in the secret	apigatewayecdsa.cer	Not applicable	
service.ssl.caBundle.k8SecretName	name of the secret which stores keys and certificates	ocudr-gateway-secret	Not applicable	
service.ssl.caBundle.k8NameSpace	namespace in which secret is created	ocudr	Not applicable	
service.ssl.caBundle.fileName	ca Bundle stored in the secret	caroot.cer	Not applicable	
service.ssl.keyStorePassword.k8SecretName	name of the secret which stores keys and certificates	ocudr-gateway-secret	Not applicable	
service.ssl.keyStorePassword.k8NameSpace	namespace in which secret is created	ocudr	Not applicable	
service.ssl.keyStorePassword.fileName	keyStore password stored in the secret	key.txt	Not applicable	
service.ssl.trustStorePassword.k8SecretName	name of the secret which stores keys and certificates	ocudr-gateway-secret	Not applicable	
service.ssl.trustStorePassword.k8NameSpace	namespace in which secret is created	ocudr	Not applicable	
service.ssl.trustStorePassword.fileName	trustStore password stored in the secret	trust.txt	Not applicable	
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	
resources.requests.cpu	Cpu allotment for ocudr-endpoint pod	3	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.memory	Memory allotment for ocudr-endpoint pod	4Gi	Not Applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	
minAvailable	Number of pods always running	2	Not Applicable	
minReplicas	Min replicas to scale to maintain an average CPU utilization	2	Not applicable	
maxReplicas	Max replicas to scale to maintain an average CPU utilization	5	Not applicable	
log.level.root	Logs to be shown on ocudr-endpoint pod	WARN	valid level	
log.level.ingress	Logs to be shown on ocudr-ingressgateway pod for ingress related flows	INFO	valid level	
log.level.oauth	Logs to be shown on ocudr-ingressgateway pod for oauth related flows	INFO	valid level	
fullnameOverride	Name to be used for deployment	ocudr-ingressgateway	Not applicable	This config is commented by default.
initssl	To Initialize SSL related infrastructure in init/update container	false	Not Applicable	
jaegerTracingEnabled	Enable/Disable Jaeger Tracing	false	true/false	
openTracing.jaeger.udpSender.host	Jaeger agent service FQDN	jaeger-agent.cne-infra	Valid FQDN	
openTracing.jaeger.udpSender.port	Jaeger agent service UDP port	6831	Valid Port	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
openTracing.jaeger.probabilisticSampler	Probablistic Sampler on Jaeger	0.5	Range: 0.0 - 1.0	Sampler makes a random sampling decision with the probability of sampling. For example, if the value set is 0.1, approximately 1 in 10 traces will be sampled
oauthValidatorEnabled	OAUTH Configuration	false	Not Applicable	
enableIncomingHttp	Enabling for accepting http requests	true	Not Applicable	
enableIncomingHttps	Enabling for accepting https requests	false	true or false	
enableOutgoingHttps	Enabling for sending https requests	false	true or false	
maxRequestsQueuedPerDestination	Queue Size at the ocudr-endpoint pod	5000	Not Applicable	
maxConnectionsPerIp	Connections from endpoint to other microServices	10	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
routesConfig	Routes configured to connect to different micro services of UDR	`- id:traffic_mapping_http uri: http://{{ .Release.Name }}-nudr-drservice:5001 path: /nudr-dr/** - id: traffic_mapping_http_prov uri: http://{{ .Release.Name }}-nudr-drservice:5001 path: /nudr-dr-prov/** - id: traffic_mapping_http_mgmt uri: http://{{ .Release.Name }}-nudr-drservice:5001 path: /nudr-dr-mgm/** - id: traffic_mapping_http_udsf uri: http://{{ .Release.Name }}-nudr-drservice:5001 path: /nudsf-dr/** - id: traffic_mapping_http_group uri: http://{{ .Release.Name }}-nudr-drservice:5001 path: /nudr-group-id-map/** - id: traffic_mapping_http_group_prov uri: http://{{ .Release.Name }}-nudr-`	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
		`drservice: 5001 path: /nudr-group-id-map-prov/**`		

4

Upgrading an Existing Unified Data Repository Deployment

To upgrade an existing UDR deployment, first upgrade the DB schema and then, perform the helm upgrade.

User should stop the Provisioning traffic while performing the upgrade procedure.

Note:

While upgrading from UDR 1.5.0 to UDR 1.6.0, user should follow the instructions in the same order as given in the **Upgrading DB Schema** section and **Upgrading Helm** section below.

Upgrading DB Schema
You should install `mysql-connector` before upgrading the DB schema.

To upgrade DB schema:

```
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/p/python2-
pip-8.1.2-7.el7.noarch.rpm
wget dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -ivh epel-release-6-8.noarch.rpm
yum install python-setuptools
rpm -ivh python2-pip-8.1.2-7.el7.noarch.rpm
pip install -U setuptools
pip install -U wheel
pip install mysql-connector-python-rf
```

Modify username, password and db name in the script as per requirement.

Note:

You can refer to the Oracle Help Center for **upgrade.py** script.

Helm Upgrade

Upgrading an existing deployment replaces the running containers and pods with new containers and pods. If there is no change in the pod configuration, it is not replaced. Unless there is a change in the service configuration of a micro service, the service endpoints remain unchanged. For example, ClusterIP.

- To upgrade, follow instructions given in the Deploying OCUDR section to extract the required OCUDR software components. If required, re-tag and push the images to customer's repository. For more information, see UDR Deployment.

- Take a backup of 1.5.0 version's **ocudr-custom-values.yaml** file before changing any configuration.
- Modify the **ocudr-custom-values-1.6.0.yaml** file parameters as per site requirement. For more information on updating the **ocudr-custom-values-1.6.0.yaml** file, see ocudr-custom-values.yaml File Configuration.

Execute the following command to upgrade an existing Unified Data Repository deployment. For the parameters that are configurable, see . Customizing Unified Data Repository

```
$ helm upgrade <release> <helm chart> [--version <OCUDR version>] -f
<ocudr-custom-values-1.6.0.yaml>

<release> could be found in the output of 'helm list' command
<chart> is the name of the chart in the form of <repository/ocudr> e.g.
reg-1/ocudr or cne-repo/ocudr
```

Rollback Instructions

Execute the following command to check if the pods are successfully started.

```
kubectl get pods -n <namespace_name>
```

If there are issues that a user cannot recover on checking logs and describe on pods, rollback using the steps below:

Schema Rollback:

1. Rollback schema to 1.5.0.
2. Use the **rollback.py** script to downgrade to 1.5.0 schema, modify username, password and db name as per requirement.
   ```
   python rollback.py
   ```

Note:

You can refer to the Oracle Help Center site for the **rollback.py** script.

Image Rollback using Helm:

1. Use the backed up customized 1.5.0 version's **ocudr-values.yaml** file to rollback to previous version.
2. Execute the helm rollback command.
   ```
   helm rollback <helm release name> <revision_no>
   ```
   To obtain the revision number, execute the following command :
   ```
   helm history <helm release name>
   ```

5
Uninstalling Unified Data Repository

To uninstall or completely delete the Unified Data Repository (UDR) deployment, execute the following command:

```
helm del --purge <helm_release_name_for_ocudr>
```

Note:

In case you are using helm3, execute the following command to uninstall UDR:

```
helm uninstall <helm_release_name_for_ocudr> --namespace
<ocudr_namespace>
```