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

Release 1.8
F34998-01
September 2020

Oracle Communications Cloud Native Unified Data Repository Installation and Upgrade Guide, Release 1.8

F34998-01

Contents

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.

What's New in This Guide

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

Release 1.8

The following new features are supported in this release:

- UDR deployment with service mesh like Aspen. This helps in controlling and monitoring the data flow within UDR microservices and outside as well.
- OAUTH2 token validation for ingress traffic.
- Bulk Import of subscriber data using import tool.
- Migration of 4G policy data from 4G UDR to 5G UDR using migration tool.
- Consolidated provisioning APIs for PCF data.
- Diameter Sh support including Quota and all other entities.
- Audit of two SLF segments using ProvGw.
- Supports UDR Alerts.
- Diameter configurations on CNC-Console GUI.

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. It is implemented as a cloud native function and offers a unified database for storing application, subscription, authentication, service authorization, policy data, session binding and Application state information. It provides a HTTP2 based RESTful interface for other 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 29.505 Release 15 specification UDM
- Is compliant to 3GPP 29.519 Release 16 (backward compatible with Release 15) specification for PCF
- Has tiered architecture providing separation between the connectivity, business logic and data layers
- Uses Oracle MySQL NDB Cluster CGE Edition as backend database in the Data Tier
- Registers with NRF in the 5G network so that the other NFs in the network can discover UDR through NRF
- Registers UDR with services like DR-SERVICE and GROUP-ID-MAP

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

Unstructured Data Storage Function (UDSF) is a part of Oracle's 5G UDR solution. It supports storage and retrieval of unstructured data by any 5G NF. The specifications of UDSF are presently not defined by 3GPP.

5G SLF functionality is also a part of Oracle's 5G UDR solution. It:

- Supports Nudr-groupid-map service as defined by 3GPP
- Registers with NRF for Nudr-groupid-map service

- Is 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 using Oauth2.
- It provides TLS support.
- It runs on Kubernetes/OCCNE as a microservice.
- It uses Egress API Gateway for Egress traffic arising from UDR (notifications and NRF management APIs).

Business Tier

- Provides the business logic of 5G Unified Data Repository.
- It has following 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.
  - **nudr-config:** Handles all request from CNC-Console and redirects all requests to appropriate REST API of the config server. It allows users to configure UDR for all micro services.
  - **nudr-config-server:** Handles all the requests from nudr-config and updates the database.
  - **nudr-diameterproxy service:** Supports Diameter Sh interface for 4G policy data for the subscriber profile.

Data Tier

- Uses Oracle MySQL NDB Cluster, CGE edition as backend database in the DB tier. This provides HA and geo-redundcancy capabilities.
- Users can build database on either Bare metal, virtualized or on kubernetes platform (kubevirt based).

References

You can refer to the following documents for better understanding of Unified Data Repository and its related network functions.

- Unified Data Repository User's Guide

- Provisioning Gateway Guide
- CNE Installation Guide
- Policy Installation Guide
- NRF Installation Guide
- Cloud Native Core Network Function Data Collector User's Guide

Acronyms

The following table provides information about the acronyms used in the document.

Field	Description
5G-AN	5G Access Network
5GC	5G Core Network
5G-GUTI	5G Globally Unique Temporary Identifier
5GS	5G System
AMF	Access and Mobility Management Function
ASM	Aspen Service Mesh
AUSF	Authentication Server Function
NEF	Network Exposure Function
NF	Network Function
NRF	Network Repository Function
NSI ID	Network Slice Instance Identifier
NSSAI	Network Slice Selection Assistance Information
NSSF	Network Slice Selection Function
NSSP	Network Slice Selection Policy
PCF	Policy Control Function
REST	Representational State Transfer
SEPP	Security Edge Protection Proxy
SLF	Subscriber Location Function
SMF	Session Management Function
UDM	Unified Data Management
UDR	Unified Data Repository
UDSF	Unstructured Data Storage Function

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	v1.17.5
HELM	v3.1.2

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

Software	Version	Notes
elasticsearch	7.6.1	Needed for Logging Area
elastic-curator	5.5.4	Needed for Logging Area
elastic-exporter	1.1.0	Needed for Logging Area
logs	3.0.0	Needed for Logging Area
kibana	7.6.1	Needed for Logging Area
grafana	6.6.2	Needed for Metrics Area
prometheus	2.16.0	Needed for Metrics Area
prometheus-node-exporter	0.18.1	Needed for Metrics Area
metallb	0.8.4	Needed for External IP
metrics-server	0.3.6	Needed for Metric Server
tracer	1.14.0	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 1.7. 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.
- **Tools Package:** It has the deployment template yaml files for nudr-migration and nudr-bulk-import services.
- **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;
   ```

   > **Note:**
   >
   > DB Name used in the above command should be same as releaseDbName configuration under global section in values.

   ```
   GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE
   TEMPORARY TABLES,
    DELETE, UPDATE, EXECUTE, INDEX, REFERENCES ON <db_name>.* TO
   '<user>'@'%';
   USE <db_name>;
   ```

   > **Note:**
   >
   > You need this database name, username and password at the time of creating Kubernetes 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
   - Keys and Certificates for oauth2 Support
5. Persistent Volume Claim Creation
6. ocudr-custom-values.yaml File Configuration
7. UDR Deployment
8. Post Installation Sanity Check - Helm Test

Installation Preparation

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

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

   Example:`ocudr-pkg-1.8.0.0.0.tgz`

2. Untar the UDR Package File. Execute the following command to untar UDR Package File.
   `tar -xvf ocudr-pkg-1.8.0.0.0.tgz`

   This command results into `ocudr-pkg-1.8.0.tgz` directory. The directory consists of following:

   - **UDR Docker Images File:** `ocudr-images-1.8.0.tar`
   - **Helm File:** `ocudr-1.8.0.tgz`
   - **Readme txt File:** The `Readme.txt` contains cksum and md5sum of tarballs.

3. Verify the checksums of tarballs in the `Readme.txt` file.

4. Load the tarballs to docker images. Execute the following command:
   `docker load --input /root/ocudr-images-1.8.0.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>`

   Sample Tag Commands:

```
docker tag ocudr/nudr_datarepository_service:1.8.0 <customer repo>/
nudr_datarepository_service:1.8.0
docker tag ocudr/nudr_nrf_client_service:1.8.0 <customer repo>/
nudr_nrf_client_service:1.8.0
docker tag ocudr/nudr_notify_service:1.8.0 <customer repo>/
nudr_notify_service:1.8.0
docker tag ocudr/nudr_diameterproxy:1.8.0 <customer repo>/
nudr_diameterproxy:1.8.0
docker tag ocudr/nudr_prehook:1.8.0 <customer repo>/
nudr_prehook:1.8.0
docker tag ocudr/nudr_bulk_import:1.8.0 <customer repo>/
nudr_bulk_import:1.8.0
docker tag ocudr/nudr_pre_upgrade_hook:1.8.0 <customer repo>/
nudr_pre_upgrade_hook:1.8.0
docker tag ocudr/nudr_migration:1.8.0 <customer repo>/
nudr_migration:1.8.0
docker tag ocudr/nudr_config:1.8.0 <customer repo>/nudr_config:1.8.0
docker tag ocudr/ocingress_gateway:1.8.1 <customer repo>/
ocingress_gateway:1.8.1
docker tag ocudr/ocegress_gateway:1.8.1 <customer repo>/
ocegress_gateway:1.8.1
```

```
docker tag ocudr/configurationinit:1.4.0 <customer repo>/
configurationinit:1.4.0
docker tag ocudr/configurationupdate:1.4.0 <customer repo>/
configurationupdate:1.4.0
docker tag ocudr/ocpm_config_server:1.7.0 <customer repo>/
ocpm_config_server:1.7.0
docker tag ocudr/readiness-detector:1.7.1 <customer repo>/readiness-
detector:1.7.1
docker tag ocudr/nf_test:1.8.0 <customer repo>/nf_test:1.8.0
```

7. Push the docker images to docker registry. Execute the following command:
   `docker push <docker-repo>/<image-name>:<image-tag>`

   Sample Push Commands:

```
docker push <customer repo>/nudr_datarepository_service:1.8.0
docker push <customer repo>/nudr_nrf_client_service:1.8.0
docker push <customer repo>/nudr_notify_service:1.8.0
docker push <customer repo>/nudr_diameterproxy:1.8.0
docker push <customer repo>/nudr_prehook:1.8.0
docker push <customer repo>/nudr_bulk_import:1.8.0
docker push <customer repo>/nudr_pre_upgrade_hook:1.8.0
docker push <customer repo>/nudr_migration:1.8.0
docker push <customer repo>/nudr_config:1.8.0
docker push <customer repo>/ocingress_gateway:1.8.1
docker push <customer repo>/ocegress_gateway:1.8.1
docker push <customer repo>/configurationinit:1.4.0
docker push <customer repo>/configurationupdate:1.4.0
docker push <customer repo>/ocpm_config_server:1.7.0
docker push <customer repo>/readiness-detector:1.7.1
docker push <customer repo>/nf_test:1.8.0
```

8. Untar Helm Files. Execute the following command:
   `tar -xvzf ocudr-1.8.0.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-1.8.0 file that contains the following:
      - **UDR_Dashboard.json:** This file is used by grafana.
      - **ocudr-custom-values-1.8.0.yaml:** This file is used during installation.
      - **ProvGw_Dashboard.json**
      - **provgw-custom-values-1.8.0.yaml**

Following are the 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
<helm_release_name>-egressgateway	ocudr/ocegress_gateway ocudr/configurationinit ocudr/configurationupdate
<helm_release_name>-nudr-config	ocudr/nudr_config
<helm_release_name>-nudr-config-server	ocudr/ocpm_config_server ocudr/readiness-detector
<helm_release_name>-nudr-diameterproxy-service	ocudr/nudr_diameterproxy
<helm_release_name>-test	ocudr/nf_test
<helm_release_name>-nudr-preinstall	ocudr/nudr_prehook
<helm_release_name>-nudr-pre-upgrade	ocudr/nudr_pre_upgrade_hook
<helm-release-name>-nudr-bulk-import	ocudr/nudr_bulk_import
<helm-release-name>-nudr-migration	ocudr/nudr_migration

Note:

<helm_release_name>-nudr-notify-service and **<helm_release_name>-nudr-diameterproxy-service** are not required for SLF deployment. So, set its flag value as 'enabled - false' in the **values.yaml** file. For more details, see User Configurable Parameterocudr-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
  - persistentvolumeclaims
  verbs:
  - get
  - watch
  - list
  - update
```

```
---

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=
   ```

   Note:

   The **name** used to define a secret above should be same as given in the **dbCredSecretName** configuration under global section in values.yaml.

   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>
```

Kubernetes Secret Creation - Keys and Certificates for Oauth2 Support

In this section, you will learn to configure Oauth Token validation and update Oauth token.

Configuring Oauth Token Validation

To configure Oauth Token Validation:

1. Create NRF access tokens using following private keys.
ECDSA private key (**Example:** `ecdsa_private_key_pkcs8.pem`)

   RSA private key (**Example:** `rsa_private_key_pkcs1.pem`)
2. To validate these access tokens, you need to create a secret and configure the public keys fetched from NRF into the ocudr-ingress-gateway. The public key naming format should be:
`<nrfInstanceId>_<AlgorithmUsed>.pem (6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c_ES256.pem`
3. To create a secret:
   a. Login to Bastion Host or a server from where you can execute kubectl.
   b. Execute the following command to create a namespace for the secret:
   `kubectl create namespace ocudr`
   c. Execute the following command to create kubernetes secret for NF access token validation:

```
kubectl create secret generic oauthsecret --from-
file=6faf1bbc-6e4a-4454-a507
-a14ef8e1bc5c_ES256.pem--from-file=6faf1bbc-6e4a-4454-a507-
a14ef8e1bc5c_RS256.pem -n
ocudr
```

> **Note:**
>
> The file names in the above command are same as in Step 1.

d. Execute the following command to verify whether sceret is created successfully:
`kubectl describe secret oauthsecret -n ocudr`

Updating Oauth Token

To update Oauth token:

1. Login to Bastion Host or a server from where you can execute kubectl.
2. To update the secret with new or updated details:
   a. Execute the following command to delete the secret and recreate it:
   `kubectl delete secret oauthsecret -n ocudr`
   b. Fetch or get the updated public keys from NRF.
   c. Execute the following command to recreate the secret with updated details:
   ```
   kubectl create secret generic oauthsecret --
   from-file=0263663c-f5c2-4d1b-9170-f7b1a9116337_ES256.pem --from-
   file=0263663c-f5c2-4d1b-9170-f7b1a9116337_RS256.pem -n ocudr
   ```

Persistent Volume Claim Creation

This step is **OPTIONAL**. When there is a need to import bulk subscribers only then you need to create a Persistent Volume Claim to use the bulk import feature.

This service is not a part of the OCUDR umbrella chart. It is a separate job, which should run in the same namespace as UDR.

The **nudr-bulk-import** service allows you to import multiple subscribers to UDR from a CSV file. To use this service, you need to create a **Persistent Volume Claim (PVC)**. To create a PVC:

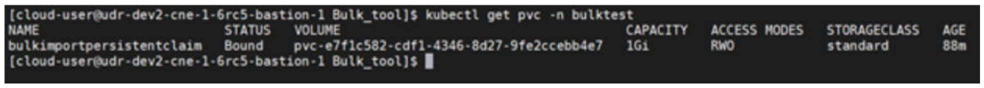
1. Use the following template to create a PVC for nudr-bulk-import service.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: bulkimportpersistentclaim
spec:
  storageClassName: <Please Provide your StorageClass  Name>
  accessModes:
    - ReadWriteOnce
```

```
resources:
  requests:
    storage: 1Gi
```

2. Save the above template to a <file-name>.yaml and execute the following command to create a Persistent Volume Claim.
   `kubectl create -f <file-name>.yaml`
3. Execute the following command to verify whether PVC is created.
   `kubectl get pvc -n <namespace>`

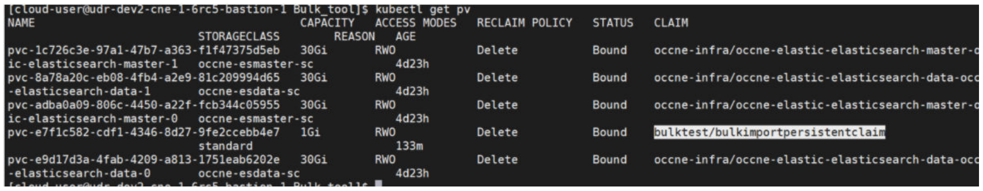
   Figure 2-1 Verifying PVC Creation

```
[cloud-user@udr-dev2-cne-1-6rc5-bastion-1 Bulk_tool]$ kubectl get pvc -n bulktest
NAME                          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
bulkimportpersistentclaim     Bound    pvc-e7f1c582-cdf1-4346-8d27-9fe2ccebb4e7   1Gi        RWO            standard       88m
[cloud-user@udr-dev2-cne-1-6rc5-bastion-1 Bulk_tool]$
```

4. Execute the following command to verify whether PV is created and bound to the PVC volume.
   `kubectl get pv`

   Figure 2-2 Verifying PV Creation

```
[cloud-user@udr-dev2-cne-1-6rc5-bastion-1 Bulk_tool]$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
                           STORAGECLASS        REASON   AGE
pvc-1c726c3e-97a1-47b7-a363-f1f47375d5eb   30Gi       RWO            Delete           Bound    occne-infra/occne-elastic-elasticsearch-master-o
ic-elasticsearch-master-1   occne-esmaster-sc            4d23h
pvc-8a78a20c-eb08-4fb4-a2e9-81c209994d65   30Gi       RWO            Delete           Bound    occne-infra/occne-elastic-elasticsearch-data-occ
-elasticsearch-data-1       occne-esdata-sc              4d23h
pvc-adba0a09-806c-4450-a22f-fcb344c05955   30Gi       RWO            Delete           Bound    occne-infra/occne-elastic-elasticsearch-master-o
ic-elasticsearch-master-0   occne-esmaster-sc            4d23h
pvc-e7f1c582-cdf1-4346-8d27-9fe2ccebb4e7   1Gi        RWO            Delete           Bound    bulktest/bulkimportpersistentclaim
                           standard                     133m
pvc-e9d17d3a-4fab-4209-a813-1751eab6202e   30Gi       RWO            Delete           Bound    occne-infra/occne-elastic-elasticsearch-data-occ
-elasticsearch-data-0       occne-esdata-sc              4d23h
```

For more details on Bulk Import Provisioning, you can refer to #unique_28.

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-1.8.0.yaml** file:

Table 2-1 ocudr-custom-values-1.8.0.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.	ocudr-registry.us.oracle.com:5000

Table 2-1 (Cont.) ocudr-custom-values-1.8.0.yaml Parameters

Section	Parameter	Services
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. • **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-1.8.0.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-1.8.0.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. All pod names, service name, deployment name are prepended by its 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-1.8.0.yaml>* - is the customized **ocudr-custom-values-1.8.0.yaml** file. The **ocudr-custom-values-1.8.0.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-1.8.0.yaml> <helm chart> [--version
> <OCUDR version>]
> ```

2. (Optional) To customize the Unified Data Repository, override 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.

Post Installation Sanity Check - Helm Test

Helm Test is a feature that validates successful installation of UDR along with its readiness (Readiness probe url configured is checked for success) of all the pods. The pods that are checked are based on the namespace and label selector configured for the helm test configurations.

This test also checks for all the PVCs to be in bound state under the Release namespace and label selector configured.

Note: You can use Helm Test feature only if you have Helm3.

To execute the Helm test functionality:

> **Note:**
>
> Before executing the Hem Test command, it is important to do the following configurations.

- Configure the helm test configurations under the Global section of the values.yaml file as follows:

```
global:
  # Helm test related configurations
  test:
    nfName: ocudr
    image:
      name: ocudr/nf_test
      tag: 1.8.0
```

```
    config:
      logLevel: WARN
      timeout: 120
```

For more details, refer to the Configuring User Parameters

- Ensure the label given below is part of all microservice deployments. The Helm Test feature takes the labelSelector internally, along with the helm release namespace, to select the pods and pvcs for verification.
  app.kubernetes.io/instance: {{ .Release.Name }}

  Usually, it is one of the Engineering labels present in the template of all NF charts. If it is not present, you need to add this label so that the helm test can work on specific helm release.
- Execute the following Helm Test command:
  `helm test <helm_release_name> -n <k8s namespace>`

  Wait for the helm test job to complete. Check the output whether the test job is successful or not.

Note:

Readiness probe for all kubernetes deployment defined under the umbrella chart should be configured with **httpGet** parameter with proper url. If it is not configured, helm test for that pod is considered success. And if the Pod/PVC list to be verified, is fetched based on namespace and labelSelector is empty, then the Helm Test is success. If the Helm Test fails with errors, then you can refer to the Troubleshooting Unified Data Repository

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.

You can prepare the **ocudr-custom-values.yaml** file 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

```
# Copyright 2019 (C), Oracle and/or its affiliates. All rights reserved.


global:

  dockerRegistry: ocudr-registry.us.oracle.com:5000



  # MYSQL Connectivity Configurations

  mysql:

    dbServiceName: "mysql-connectivity-service.occne-infra"  #This is a 
read only parameter. Use the default value.

    port: "3306"



  # Jaeger tracing Configurations
```

```
udrTracing:
  enable: false
  host: "occne-tracer-jaeger-collector.occne-infra"
  port: 14268

dbenc:
  shavalue: 256

# Configure customer created service accounts
serviceAccountName:

# Configuration to enable UDR egress traffic through EGW
egress:
  enabled: true

# Config server related configurations
configServerEnable: true
initContainerEnable: false
dbCredSecretName: 'ocudr-secrets'
configServerFullNameOverride: nudr-config-server

# Configuration to decide the Service the deployment will provide
udrServices: "All"

# Enable to register with NRF for UDSF service
udsfEnable: false
```

```
  # port on which UDR's API-Gateway service is exposed

  # If httpsEnabled is false, this Port would be HTTP/2.0 Port
(unsecured)

  # If httpsEnabled is true, this Port would be HTTPS/2.0 Port (secured
SSL)

  publicHttpSignalingPort: 80

  publicHttpsSignallingPort: 443



  # Nf Instance ID for UDR, same will be registered with NRF

  nfInstanceId: 5a7bd676-ceeb-44bb-95e0-f6a55a328b03



  # Helm test hook related configurations

  test:

    nfName: ocudr

    image:

      name: ocudr/nf_test

      tag: 1.8.0

    config:

      logLevel: WARN

      timeout: 120      #Beyond this duration helm test will be
considered failure



  # Pre Install Hook configurations. Used for DB Creation

  preInstall:

    image:

      name: ocudr/nudr_prehook

      tag: 1.8.0

    config:

      logLevel: WARN
```

```
# Pre Upgrade Hook configurations. Used for DB Schema Upgrade

  preUpgrade:

    image:

      name: ocudr/nudr_pre_upgrade_hook

      tag: 1.8.0

    config:

      logLevel: WARN


  # Resource allocation for all UDR hooks

  hookJobResources:

    limits:

      cpu: 2

      memory: 2Gi

    requests:

      cpu: 1

      memory: 1Gi



#***********************************************************************
***



  # ********  Sub-Section Start: Custom Extension Global Parameters
********


#***********************************************************************
***



  customExtension:

    # Applicable for all resources created as part of helm intallation
    allResources:

      labels: {}

      annotations: {}


    # Applicable for all load balancer type services

    lbServices:

      labels: {}

      annotations: {}


    # Applicable for all load balancer type deployments

    lbDeployments:

      labels: {}

      annotations: {}


    # Applicable for all non load balancer type services

    nonlbServices:

      labels: {}

      annotations: {}


    # Applicable for all non load balancer type deployments

    nonlbDeployments:

      labels: {}

      annotations: {}


  # ********  Sub-Section End: Custiom Extensions Global Parameters
********


#**************************************************************************
***
```

```
  # ********  Sub-Section Start: Prefix/Suffix Global Parameters
************

#*************************************************************************
***


  k8sResource:

    container:

      prefix:

      suffix:


  # ********  Sub-Section End: Prefix/Suffix Global Parameters
*************

#*************************************************************************
***


# nudr-drservice microservice configurations

nudr-drservice:

#  nameOverride: "nudr-drservice"

#  Image Details

  image:

    name: ocudr/nudr_datarepository_service

    tag: 1.8.0

    pullPolicy: Always


  service:

    # Enable http2 server

    http2enabled: "true"

    # k8s Service type
    type: ClusterIP

    # Ports used in dr service. Applicable for both container and
service ports.

    port:

      http: 5001

      https: 5002

      management: 9000

    # Microservice specific annotation for exposed service

    customExtension:

      labels: {}

      annotations: {}

  # Flag to enable/disable dr service tracing

  tracingEnabled: false



  # nudr-notify service ports used. Should be same as the ports
configured under nudr-notify-service section

  notify:

    port:

      http: 5001

      https: 5002



  deployment:

    # Replica count for deployment

    replicaCount: 2

    # Microservice specific annotation for deployment

    customExtension:

      labels: {}

      annotations: {}
  # Logging level

  logging:

    level:

      root: "WARN"


  # Flag to enable/disable autocreation of subscriber when we do PUT
operataion on a new UEID

  subscriber:

    autocreate: "true"


  # Flag to validate smdata

  validate:

    smdata: "false"


  # Decides where the vsaLevel parameter will be placed in the data

  vsaLevel: "smpolicy" # sample values {"smpolicy" or "nssai" or "dnn"}

  vsaBillingDay: 0


  # Resource specification for nudr-drservice container

  resources:

    limits:

      cpu: 4

      memory: 4Gi

    requests:

      cpu: 4

      memory: 4Gi
    # When CPU utilization goes beyond this limit, new pod will be
scaled by HPA

    target:

      averageCpuUtil: 80



  # MYSQL connection pool size

  hikari:

    poolsize: "25"



  # Minumum replica count to be maintaned by HPA. Suggested to keep
same as deployment.replicaCount

  minReplicas: 2

  # Maximum replicas that can be scaled by HPA

  maxReplicas: 8



  # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

  # should consider tuning these parameters.

  readinessProbe:

    # tells the kubelet that it should wait second before performing
the first probe

    initialDelaySeconds: 70

    # specifies that the kubelet should perform a readiness probe every
xx seconds

    periodSeconds: 10



  # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

  # should consider tuning these parameters.

  livenessProbe:

    # tells the kubelet that it should wait second before performing
```

```
the first probe

    initialDelaySeconds: 70

    # specifies that the kubelet should perform a liveness probe every
xx seconds

    periodSeconds: 10



# nudr-notify-service microservice configurations

nudr-notify-service:

#  nameOverride: "nudr-notify-service"

#  Enable/Disable nudr-notify-service deployment

  enabled: true



  # Image Details

  image:

    name: ocudr/nudr_notify_service

    tag: 1.8.0

    pullPolicy: Always



  service:

    # Enable http2 server

    http2enabled: "true"

    # k8s Service type

    type: ClusterIP

    # Ports used in notify service. Applicable for both container and
service ports.

    port:

      http: 5001

      https: 5002

      management: 9000
    # Microservice specific annotation for exposed service
    customExtension:
      labels: {}
      annotations: {}

  # Flag to enable/disable dr service tracing
  tracingEnabled: false

  deployment:
    # Replica count for deployment
    replicaCount: 2
    # Microservice specific annotation for deployment
    customExtension:
      labels: {}
      annotations: {}

  notification:
    # Retry count for failed notifications
    retrycount: "3"
    # Interval for each retry attempt
    retryinterval: "5"
    # Error codes for which notification will be retried
    retryerrorcodes: "400,429,500,503"

  # MYSQL connection pool size
  hikari:
    poolsize: "10"
  # Logging level
  logging:
    level:
      root: "WARN"

  # Resource specification for nudr-notify-service container
  resources:
    limits:
      cpu: 3
      memory: 3Gi
    requests:
      cpu: 3
      memory: 3Gi

    # When CPU utilization goes beyond this limit, new pod will be 
scaled by HPA
    target:
      averageCpuUtil: 80

  # Minumum replica count to be maintaned by HPA. Suggested to keep 
same as deployment.replicaCount
  minReplicas: 2
  # Maximum replicas that can be scaled by HPA
  maxReplicas: 4

  # Egress Gateway port to be used for connection
  http:
    proxy:
      port: 8080


  # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

  # should consider tuning these parameters.

  readinessProbe:

    # tells the kubelet that it should wait second before performing
the first probe

    initialDelaySeconds: 80

    # specifies that the kubelet should perform a readiness probe every
xx seconds

    periodSeconds: 5



  # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

  # should consider tuning these parameters.

  livenessProbe:

    # tells the kubelet that it should wait second before performing
the first probe

    initialDelaySeconds: 80

    # specifies that the kubelet should perform a liveness probe every
xx seconds

    periodSeconds: 20



nudr-config:

#  nameOverride: "nudr-configuration-service"

#  Enable/Disable nudr-config deployment

  enabled: true



  # Image Details
  image:

    name: ocudr/nudr_config

    tag: 1.8.0

    pullPolicy: Always


  service:

    # Enable http2 server

    http2enabled: "true"

    # k8s Service type

    type: ClusterIP

    #Ports used in nudr-config service. Applicable for both container
and service ports.

    port:

      http: 5001

      https: 5002

      management: 9000

    # Microservice specific annotation for exposed service

    customExtension:

      labels: {}

      annotations: {}


  deployment:

    # Replica count for deployment

    replicaCount: 1

    # Microservice specific annotation for deployment

    customExtension:

      labels: {}

      annotations: {}
  # Logging level

  logging:

    level:

      root: "WARN"


  # Resource specification for nudr-config container

  resources:

    limits:

      cpu: 2

      memory: 2Gi

    requests:

      cpu: 2

      memory: 2Gi


    # When CPU utilization goes beyond this limit, new pod will be 
scaled by HPA

    target:

      averageCpuUtil: 80


  # Minumum replica count to be maintaned by HPA. Suggested to keep 
same as deployment.replicaCount

  minReplicas: 1

  # Maximum replicas that can be scaled by HPA

  maxReplicas: 1


  # Do not change any values in this section. If we see delays in pod 
coming up and probe is killing the pod then we

  # should consider tuning these parameters.
  readinessProbe:

    # tells the kubelet that it should wait second before performing 
the first probe

    initialDelaySeconds: 30

    # specifies that the kubelet should perform a readiness probe every 
xx seconds

    periodSeconds: 5



  # Do not change any values in this section. If we see delays in pod 
coming up and probe is killing the pod then we

  # should consider tuning these parameters.

  livenessProbe:

    # tells the kubelet that it should wait second before performing 
the first probe

    initialDelaySeconds: 40

    # specifies that the kubelet should perform a liveness probe every 
xx seconds

    periodSeconds: 40



# config-server related configurations

config-server:

  # Enable/Disable config-server deployment

  enabled: true



  global:

    nfName: nudr

    # Init service image to be used if global.initContainerEnable is 
set to true

    imageServiceDetector: ocudr/readiness-detector:1.7.1

    # Jaeger configurations for Config-server tracing

    envJaegerAgentHost: ''
    envJaegerAgentPort: 6831

  replicas: 1
  envLoggingLevelApp: WARN

  # Resource specification for nudr-drservice container
  resources:
    limits:
      cpu: 2
      memory: 2Gi
    requests:
      cpu: 2
      memory: 512Mi

  service:
    # k8s Service type
    type: ClusterIP
    port: 0
    # Microservice specific annotation for exposed service
    customExtension:
      labels: {}
      annotations: {}

  deployment:
    # Microservice specific annotation for deployment
    customExtension:
      labels: {}
```

```
        annotations: {}



    fullnameOverride: udr-config-server

    installedChartVersion: ''



    # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

    # should consider tuning these parameters.

    readinessProbe:

      # tells the kubelet that it should wait second before performing
the first probe

      initialDelaySeconds: 20

      # Number of seconds after which the probe times out

      timeoutSeconds: 3

      # specifies that the kubelet should perform a readiness probe every
xx seconds

      periodSeconds: 10

      # Minimum consecutive successes for the probe to be considered
successful after having failed

      successThreshold: 1

      # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

      failureThreshold: 3



    # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

    # should consider tuning these parameters.

    livenessProbe:

      # tells the kubelet that it should wait second before performing
the first probe

      initialDelaySeconds: 60
    # Number of seconds after which the probe times out

    timeoutSeconds: 3

    # specifies that the kubelet should perform a liveness probe every
xx seconds

    periodSeconds: 15

    # Minimum consecutive successes for the probe to be considered
successful after having failed

    successThreshold: 1

    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

    failureThreshold: 3



# nudr-nrf-client-service related configurations

nudr-nrf-client-service:

#  nameOverride: "nudr-nrf-client-service"

#  Enable/Disable nudr-notify-service deployment

  enabled: true

  # NRF ingressgateway details along with registration url and proxy
config if any

  host:

    baseurl: "http://ocnrf-ingressgateway.mynrf.svc.cluster.local/nnrf-
nfm/v1/nf-instances"

    proxy:

  # Enable SSL for nrf client service

  ssl: "false"



  # Logging level config

  logging:

    level:

      root: "WARN"
  # Image details

  image:

    name: ocudr/nudr_nrf_client_service

    tag: 1.8.0

    pullPolicy: Always



  # Heart beat timer for Update NF Profile requests to NRF

  heartBeatTimer: "90"

  # UDR group id sent in NF Profile

  udrGroupId: "udr-1"

  #  Capacity multiplier of UDR based on number of dr service UDR pods
running

  capacityMultiplier: "500"

  # Supported SUPI range registered with NRF

  supirange: "[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]"

  # Priority parameter in Nf Profile

  priority: "10"

  # IPV4 address of UDR used in registration

  udrMasterIpv4: "10.0.0.0"

  # Supported GPSI range registered with NRF

  gpsirange: "[{\"start\": \"10000000000\", \"end\": \"20000000000\"}]"

  #endpointLabelSelector : "ocudr-ingressgateway"

  # Supported plmn values for the UDR

  plmnvalues: "[{\"mnc\": \"14\", \"mcc\": \"310\"}]"

  # Client scheme used for all egress messages

  scheme: "http"

  # Liveness check retry attempts on failure
  livenessProbeMaxRetry: 5

  # this is for egress port

  http:

    proxy:

      host:

      port: 8080

 # 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"



  # Resource specification for nudr-nrf-client-service container

  resources:

    limits:

      cpu: 1

      memory: 2Gi

    requests:

      cpu: 1

      memory: 2Gi



  service:

    customExtension:

      labels: {}

      annotations: {}



  deployment:

    # Microservice specific annotation for deployment
    customExtension:
      labels: {}
      annotations: {}

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
   staticNodePortEnabled: false

   # In case of ASPEN Service Mesh enabled, to support clear text
traffic from outside of the cluster below flag needs to be true.
   istioIngressTlsSupport:
     ingressGateway: false
```

```
image:
  # image name
  name: ocudr/ocingress_gateway
  # tag name of image
  tag: 1.8.1
  # 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: 1.4.0
  # 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: 1.4.0
  # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
  pullPolicy: Always

deployment:
  # Microservice specific annotation for deployment
  customExtension:
     labels: {}
```

```
      annotations: {}

 service:
   # Microservice specific annotation for service exposed
   customExtension:
      labels: {}
      annotations: {}
   # Configure this section to support TLS with ingress gateway
   ssl:
     # TLS verison used
     tlsVersion: TLSv1.2

     # Secret Details for certificates
     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

 # This section default values can be retained. USed to support HTTP1.1
to ingressgateway
 cncc:
   enabled: false
   enablehttp1: true

 # Resource details for IGW, init and also update containers
 resources:
   limits:
     cpu: 5

     memory: 4Gi

     initServiceCpu: 1

     initServiceMemory: 1Gi

     updateServiceCpu: 1

     updateServiceMemory: 1Gi

   requests:

     cpu: 5

     memory: 4Gi

     initServiceCpu: 1

     initServiceMemory: 1Gi

     updateServiceCpu: 1

     updateServiceMemory: 1Gi

   # When CPU utilization goes beyond this limit, new pod will be
scaled by HPA

   target:

     averageCpuUtil: 80



 # Logging level

 log:

   level:

     root: WARN

     ingress: INFO

     oauth: INFO



 # enable jaeger tracing

 jaegerTracingEnabled: false
```

```
 openTracing :

   jaeger:

     udpSender:

       # udpsender host

       host: "occne-tracer-jaeger-agent.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



 # Do not change any values in this section. If we see delays in pod 
coming up and probe is killing the pod then we

 # should consider tuning these parameters.

 readinessProbe:

   # tells the kubelet that it should wait second before performing the 
first probe

   initialDelaySeconds: 30

   # Number of seconds after which the probe times out

   timeoutSeconds: 3

   # specifies that the kubelet should perform a readiness probe every 
xx seconds

   periodSeconds: 10

   # Minimum consecutive successes for the probe to be considered 
successful after having failed
```

```
   successThreshold: 1

   # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

   failureThreshold: 3



 # Do not change any values in this section. If we see delays in pod
coming up and probe is killing the pod then we

 # should consider tuning these parameters.

 livenessProbe:

   # tells the kubelet that it should wait second before performing the
first probe

   initialDelaySeconds: 30

   # Number of seconds after which the probe times out

   timeoutSeconds: 3

   # specifies that the kubelet should perform a liveness probe every
xx seconds

   periodSeconds: 15

   # Minimum consecutive successes for the probe to be considered
successful after having failed

   successThreshold: 1

   # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

   failureThreshold: 3



 # 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: UDR
#Moved to global section
#nfInstanceId: 5a7bd676-ceeb-44bb-95e0-f6a55a328b03
producerScope: nudr-dr,nudr-group-id-map
allowedClockSkewSeconds: 0
nrfPublicKeyKubeSecret: oauthsecret
nrfPublicKeyKubeNamespace: ocudr
validationType: strict
producerPlmnMNC: 14
producerPlmnMCC: 310

#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/**

  order: 1

- id: traffic_mapping_http_prov

  uri: http://{{ .Release.Name }}-nudr-drservice:5001

  path: /nudr-dr-prov/**

  order: 2

- id: traffic_mapping_http_mgmt

  uri: http://{{ .Release.Name }}-nudr-drservice:5001

  path: /nudr-dr-mgm/**

  order: 3

- id: traffic_mapping_http_udsf

  uri: http://{{ .Release.Name }}-nudr-drservice:5001

  path: /nudsf-dr/**

  order: 4

- id: traffic_mapping_http_group
```

```
    uri: http://{{ .Release.Name }}-nudr-drservice:5001
    path: /nudr-group-id-map/**
    order: 5
  - id: traffic_mapping_http_group_prov
    uri: http://{{ .Release.Name }}-nudr-drservice:5001
    path: /nudr-group-id-map-prov/**
    order: 6
  - id: traffic_mapping_http_slf_group_prov
    uri: http://{{ .Release.Name }}-nudr-drservice:5001
    path: /slf-group-prov/**
    order: 7

egressgateway:
  enabled: true
  #fullnameOverride : 'ocudr-egress-gateway'
  nfType: UDR

  #global:
  #  dockerRegistry: reg-1:5000

  deploymentEgressGateway:
    image: ocudr/ocegress_gateway
    imageTag: 1.8.1
    pullPolicy: Always

  initContainersImage:
    # inint Containers image name
    name: configurationinit
    # tag name of init Container image
    tag: 1.4.0
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  updateContainersImage:
    # update Containers image name
    name: configurationupdate
    # tag name of update Container image
    tag: 1.4.0
    # Pull Policy - Possible Values are:- Always, IfNotPresent, Never
    pullPolicy: Always

  # enable jagger tracing
  jaegerTracingEnabled: false

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

  # ---- Oauth Configuration - BEGIN ----
  oauthClient:
    enabled: false
    dnsSrvEnabled: false
    httpsEnabled: false
    virtualFqdn: localhost:port
    staticNrfList:
      - localhost:port
    nfType: UDR
    #Moved to global section
    #nfInstanceId: 5a7bd676-ceeb-44bb-95e0-f6a55a328b03
    consumerPlmnMNC: 14
    consumerPlmnMCC: 310
    maxRetry: 2
    apiPrefix: ""
    errorCodeSeries: 4XX
    retryAfter: 5000
  # ---- Oauth Configuration - END ----

  #jetty client configuration
  maxConcurrentPushedStreams: 1000
  maxRequestsQueuedPerDestination: 1024
  #maxConnectionsPerDestination: 4
  maxConnectionsPerIp: 4
  connectionTimeout: 10000 #(ms)
  requestTimeout: 1000 #(ms)
  jettyIdleTimeout: 0 #(ms,<=0 -> to make timeout infinite)

  minReplicas: 1
  maxReplicas: 4
```

```
  minAvailable: 1


  # ---- HTTPS Configuration - BEGIN ----

  initssl: false

  enableOutgoingHttps: false


  # Resource details for EGW, init and update container

  resources:

    limits:

      cpu: 3

      memory: 4Gi

      initServiceCpu: 1

      initServiceMemory: 1Gi

      updateServiceCpu: 1

      updateServiceMemory: 1Gi

    requests:

      cpu: 3

      memory: 4Gi

      initServiceCpu: 1

      initServiceMemory: 1Gi

      updateServiceCpu: 1

      updateServiceMemory: 1Gi

    # When CPU utilization goes beyond this limit, new pod will be
scaled by HPA

    target:

      averageCpuUtil: 80


  deployment:
    # Microservice specific annotation for deployment

    customExtension:

      labels: {}

      annotations: {}


  service:

    type: ClusterIP

    # Microservice specific annotation for service

    customExtension:

      labels: {}

      annotations: {}


    # This section needs to be configured to support TLS on
ingressgateway

    ssl:

      tlsVersion: TLSv1.2

      initialAlgorithm: RSA256


      # Secret related info for certificates

      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
# ---- HTTPS Configuration - END ----

#Enable this if loadbalancing is to be done by egress instead of K8s
K8ServiceCheck: false

#Set the root log level
```

```
  log:

    level:

      root: WARN

      egress: INFO

      oauth: INFO



  readinessProbe:

    # tells the kubelet that it should wait second before performing
the first probe

    initialDelaySeconds: 30

    # Number of seconds after which the probe times out

    timeoutSeconds: 3

    # specifies that the kubelet should perform a readiness probe every
xx seconds

    periodSeconds: 10

    # Minimum consecutive successes for the probe to be considered
successful after having failed

    successThreshold: 1

    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

    failureThreshold: 3



  livenessProbe:

    # tells the kubelet that it should wait second before performing
the first probe

    initialDelaySeconds: 30

    # Number of seconds after which the probe times out

    timeoutSeconds: 3

    # specifies that the kubelet should perform a liveness probe every
xx seconds

    periodSeconds: 15
    # Minimum consecutive successes for the probe to be considered
successful after having failed

    successThreshold: 1

    # When a Pod starts and the probe fails, Kubernetes will try
failureThreshold times before giving up

    failureThreshold: 3



nudr-diameterproxy:

  #  Enable/Disable nudr-diameterproxy deployment

  enabled: true



  # Image Details

  image:

    name: ocudr/nudr_diameterproxy

    tag: 1.8.0

    pullPolicy: Always



  service:

    # Enable http2 rest server

    http2enabled: "true"

    # K8s service type

    type: ClusterIP

    # K8s service type for Diameter endpoint

    diameter:

      type: LoadBalancer



    # Ports used in diameterproxy service. Applicable for both
container and service ports.

    port:
```

```
    http: 5001
    https: 5002
    management: 9000
    diameter: 6000
  # Microservice specific annotation for exposed service
  customExtension:
    labels: {}
    annotations: {}

deployment:
  # Replica count for deployment
  replicaCount: 2
  # Microservice specific annotation for deployment
  customExtension:
    labels: {}
    annotations: {}

# Logging level
logging:
  level:
    root: "WARN"

# Resource specification for nudr-diameterproxy container
resources:
  limits:
    cpu: 3
    memory: 4Gi
    requests:

      cpu: 3

      memory: 4Gi

    # When CPU utilization goes beyond this limit, new pod will be
scaled by HPA

    target:

      averageCpuUtil: 80



  # Minumum replica count to be maintaned by HPA. Suggested to keep
same as deployment.replicaCount

  minReplicas: 2

  # Maximum replicas that can be scaled by HPA

  maxReplicas: 4



  # nudr-drservice port details. Should be

  drservice:

    port:

      http: 5001

      https: 5002



  diameter:

    # Host realm of diameterproxy

    realm: "oracle.com"

    # Host realm of diameterproxy

    identity: "nudr.oracle.com"

    IO:

      # Number of threads for IO operation

      threadCount: 0      # should not go beyond 2*CPU

      # Queue size for IO
```

```
          queueSize: 0        # range [2048-8192] should be power of 2

        messageBuffer:

          # Number of threads for processing the message

          threadCount: 0      # should not go beyond 2*CPU

          # Queue Size for message processing

          queueSize: 0        # range [1024-4096] and default 1024/Low,
    2048/Medium, 4096/High. should be power of 2

        # Diameter peer setting, Parameter details below

        # reconnect delay for diameter reconnect (in seconds)

        # total turnaround time for process the diameter messages.(in sec)

        # TCP connection timeout time.(in sec)

        # DWR and DWA messages every number of time (in sec)

        # Transport layer

        # reconnect the number of time if diameter peer is down

        peer:

          setting: |

             reconnectDelay: 3

             responseTimeout: 4

             connectionTimeOut: 3

             watchdogInterval: 6

             transport: 'TCP'

             reconnectLimit: 50

          # Diameter server peer node information

          # The below information should be yaml list

          nodes: |

           - name: 'seagull'

             responseOnly: false

             namespace: 'seagull1'
             host: '10.75.185.158'

             domain: 'svc.cluster.local'

             port: 4096

             realm: 'seagull1.com'

             identity: 'seagull1a.seagull1.com'

          # Diameter client node information

          # The below information should be yaml list

          clientNodes: |

           - identity: 'seagull1a.seagull1.com'

             realm: 'seagull1.com'

           - identity: 'seagull1.com'

             realm: 'seagull1.com'


      # Do not change any values in this section. If we see delays in pod 
    coming up and probe is killing the pod then we

      # should consider tuning these parameters.

      readinessProbe:

        # tells the kubelet that it should wait second before performing 
    the first probe

        initialDelaySeconds: 80

        # specifies that the kubelet should perform a readiness probe every 
    xx seconds

        periodSeconds: 5



      # Do not change any values in this section. If we see delays in pod 
    coming up and probe is killing the pod then we

      # should consider tuning these parameters.

      livenessProbe:

        # tells the kubelet that it should wait second before performing 
    the first probe
```

```
    initialDelaySeconds: 80

    # specifies that the kubelet should perform a liveness probe every
xx seconds

    periodSeconds: 20
```

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

Global Configuration: These values are suffixed to all the container names of OCUDR. These values are useful to add custom annotation(s) to all non-Load Balancer Type Services that OCUDR helm chart creates.

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	ocudr-registry.us.oracle.com: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 for DB Service Connection	3306	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
udrTracing.enable	Flag to enable udr tracing on Jaeger	false	true/false	
udrTracing.host	Jaegar Service Name installed in CNE	occne-tracer-jaeger-collector.occne-infra	Not applicable	
udrTracing.port	Jaegar Service Port installed in CNE	14268	Not applicable	
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.
egress.enabled	Flag to enable outgoing traffic through egress gateway	true	true/false	
configServerEnable	Flag to enable config-server	true	true/false	
initContainerEnable	Flag to disable init container for config-server. This is not required because the pre install hooks take care of DB tables creation and connectivity is also verified	false	true/false	
dbCredSecretName	DB Credentioal Secret Name	ocudr-secrets	Not Applicable	
configServerFullNameOverride	Config Server Full Name Override	nudr-config-server	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
udrServices	Services supported on the UDR deployment, This config decides the schema execution on the udrdb which is done by the nudr-preinstall hook pod.	All	All/nudr-dr/nudr-group-id-map	For SLF, set udrServices values as nudr-group-id-map.
udsfEnable	Flag to enable UDSF services on the deployment	false	true/false	
publicHttpSignalingPort	Port on which ingressgateway listens for incoming http requests.	80	Valid Port	
publicHttpsSignallingPort	Port on which ingressgateway listens for incoming https requests.	443	Valid Port	
nfInstanceId	Nf Instance ID for UDR (same is registered with NRF)	5a7bd676-ceeb-44bb-95e0-f6a55a328b03	Valid uuid	A valid UUID is a 128-bit unique number that helps to identify information in computer systems.
test.nfName	NF name on which the helm test is performed. For UDR the default value is UDR. Will be used in container name as suffix	ocudr	Not applicable	
test.image.name	Image name for the helm test container image	ocudr/nf_test	Not Applicable	
test.image.tag	Image version tag for helm test	1.8.0	Not Applicable	
test.config.logLevel	Log level for helm test pod	WARN	Possible Values - WARN INFO DEBUG	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
test.config.timeout	Timeout value for the helm test operation. If exceeded helm test will be considered as failure	120	Range: 1-300 Unit:seconds	
preinstall.image.name	Image name for the nudr-prehook pod which will take care of DB and table creation for UDR deployment.	ocudr/prehook	Not Applicable	
preinstall.image.tag	Image version for nudr-prehook pod image	1.8.0	Not Applicable	
preinstall.config.logLevel	Log level for preinstall hook pod	WARN	Possible Values - WARN INFO DEBUG	
hookJobResources.limits.cpu	CPU limit for pods created kubernetes hooks/jobs created as part of UDR installation. Applicable for helm test job as well.	2	Not Applicable	
hookJobResources.limits.memory	Memory limit for pods created kubernetes hooks/jobs created as part of UDR installation. Applicable for helm test job as well.	2Gi	Not Applicable	
hookJobResources.requests.cpu	CPU requests for pods created kubernetes hooks/jobs created as part of UDR installation. Applicable for helm test job as well.	1	Not Applicable	The cpu to be allocated for hooks during deployment

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
hookJobResources.requests.memory	Memory requests for pods created k8s hooks/jobs created as part of UDR installation. Applicable for helm test job as well.	1Gi	Not Applicable	The memory to be allocated for hooks during deployment
customExtension.allResources.labels	Custom Labels that needs to be added to all the OCUDR kubernetes resources	null	Not Applicable	This can be used to add custom label(s) to all k8s resources that will be created by OCUDR helm chart.
customExtension.allResources.annotations	Custom Annotations that needs to be added to all the OCUDR kubernetes resources	null	Not Applicable **Note:** ASM related annotations needs to be added under ASM Specific Configuration section	This can be used to add custom annotation(s) to all k8s resources that will be created by OCUDR helm chart.
customExtension.lbServices.labels	Custom Labels that needs to be added to OCUDR Services that are considered as Load Balancer type	null	Not Applicable	This can be used to add custom label(s) to all Load Balancer Type Services that will be created by OCUDR helm chart.
customExtension.lbServices.annotations	Custom Annotations that needs to be added to OCUDR Services that are considered as Load Balancer type	null	Not Applicable	This can be used to add custom annotation(s) to all Load Balancer Type Services that will be created by OCUDR helm chart.
customExtension.lbDeployments.labels	Custom Labels that needs to be added to OCUDR Deployments that are associated to a Service which is of Load Balancer type	null	Not Applicable	This can be used to add custom label(s) to all Deployments that will be created by OCUDR helm chart which are associated to a Service which if of Load Balancer Type.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
customExtension .lbDeployments.a nnotations	Custom Annotations that needs to be added to OCUDR Deployments that are associated to a Service which is of Load Balancer type	null	Not Applicable **Note:** ASM related annotations needs to be added under ASM Specific Configuration section	This can be used to add custom annotation(s) to all Deployments that will be created by OCUDR helm chart which are associated to a Service which if of Load Balancer Type.
customExtension .nonlbServices.la bels	Custom Labels that needs to be added to OCUDR Services that are considered as not Load Balancer type	null	Not Applicable	This can be used to add custom label(s) to all non-Load Balancer Type Services that will be created by OCUDR helm chart.
customExtension .nonlbServices.a nnotations	Custom Annotations that needs to be added to OCUDR Services that are considered as not Load Balancer type	null	Not Applicable	This can be used to add custom annotation(s) to all non-Load Balancer Type Services that will be created by OCUDR helm chart.
customExtension .nonlbDeploymen ts.labels	Custom Labels that needs to be added to OCUDR Deployments that are associated to a Service which is not of Load Balancer type	null	Not Applicable	This can be used to add custom label(s) to all Deployments that will be created by OCUDR helm chart which are associated to a Service which if not of Load Balancer Type.
customExtension .nonlbDeploymen ts.annotations	Custom Annotations that needs to be added to OCUDR Deployments that are associated to a Service which is not of Load Balancer type	null	Not Applicable **Note:** ASM related annotations to be added under ASM Specific Configuration section	This can be used to add custom annotation(s) to all Deployments that will be created by OCUDR helm chart which are associated to a Service which if not of Load Balancer Type.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
k8sResource.container.prefix	Value that will be prefixed to all the container names of OCUDR.	null	Not Applicable	This value will be used to prefix to all the container names of OCUDR.
k8sResource.container.suffix	Value that will be suffixed to all the container names of OCUDR.	null	Not Applicable	This value will be used to prefix to all the container names of OCUDR.

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	1.8.0	Not applicable	
image.pullPolicy	This setting signifies whether image needs to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
subscriber.autocreate	Flag to enable auto creation of subscriber	true	true/false	This flag enables 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 controls 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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
maxReplicas	Maximum Replicas	8	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
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	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
hikari.poolsize	Mysql Connection pool size	25	Not applicable	The hikari pool connection size to be created at start up
vsaLevel	The data level where the vsa which holds the 4G Policy data is added.	smpolicy	Not applicable	
vsaBillingDay	The Billing day value	0	Not applicable	
tracingEnabled	Flag to enable/ disable jaeger tracing for nudr-drservice	false	true/false	
service.customExtension.labels	Custom Labels that needs to be added to nudr-drservice specific Service.	null	Not Applicable	This can be used to add custom label(s) to nudr-drservice Service.
service.customExtension.annotations	Custom Annotations that needs to be added to nudr-drservice specific Services.	null	Not Applicable	This can be used to add custom annotation(s) to nudr-drservice Service.
deployment.customExtension.labels	Custom Labels that needs to be added to nudr-drservice specific deployment.	null	Not Applicable	This can be used to add custom label(s) to nudr-drservice Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to nudr-drservice specific deployment.	null	Not Applicable	This can be used to add custom annotation(s) to nudr-drservice deployment.
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	70	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.periodSeconds	Time interval for every readiness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	70	Not Applicable Unit: Seconds	
livenessProbe.periodSeconds	Time interval for every liveness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	

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.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
image.name	Docker Image name	ocudr/ nudr_notify_servi ce	Not applicable	
image.tag	Tag of Image	1.8.0	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
notification.retryc ount	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.retrye rrorcodes configuration.
notification.retryin terval		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.retrye rrorcodes configuration.
notification.retrye rrorcodes	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.
hikari.poolsize	Mysql Connection pool size	10	Not applicable	The hikari pool connection size to be created at start up
tracingEnabled	Flag to enable/ disable jaeger tracing for nudr- notify-service	false	true/false	
http.proxy.port	Port to connect to egress gateway	8080	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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
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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.memory	Memory allotment for nudr-notify-service pod	3Gi	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	3Gi	Not applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
service.customExtension.labels	Custom Labels that needs to be added to nudr-notify-service specific service.	null	Not Applicable	This can be used to add custom label(s) tonudr-notify-service Service.
service.customExtension.annotations	Custom Annotations that needs to be added to nudr-notify-service specific services.	null	Not Applicable	This can be used to add custom annotation(s) to nudr-notify-service Service.
deployment.customExtension.labels	Custom Labels that needs to be added to nudr-notify-service specific deployment.	null	Not Applicable	This can be used to add custom label(s) to nudr-notify-service deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to nudr-notify-service specific deployment.	null	Not Applicable	This can be used to add custom annotation(s) to nudr-notify-service deployment.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	80	Not Applicable Unit: Seconds	
readinessProbe.periodSeconds	Time interval for every readiness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	5	Not Applicable Unit: Seconds	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	80	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbe.periodSeconds	Time interval for every liveness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	20	Not Applicable Unit: Seconds	

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	1.8.0	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	
udrGroupId	Group ID of UDR	udr-1	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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
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	
livenessProbeMaxRetry	Max retries of liveness proble failed	5	This should be changed based on how many times do you want to retry	This should be changed based on how many times do you want to retry if liveness fails
udrMasterIpv4	Master IP of which we deployed	10.0.0.0	This should be changed with the master ip which we deployed	udrMasterIpv4 is used to send the ipv4 address to the nrf while registration.
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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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	
http.proxy.port	Port to connect egress gateway	8080	Not applicable	
service.customExtension.labels	Custom Labels that needs to be added to nudr-nrf-client specific service.	null	Not Applicable	This can be used to add custom label(s) to nudr-nrf-client service.
service.customExtension.annotations	Custom Annotations that needs to be added to nudr-nrf-client specific services.	null	Not Applicable	This can be used to add custom annotation(s) to nudr-nrf-client service.
deployment.customExtension.labels	Custom Labels that needs to be added to nudr-nrf-client specific deployment.	null	Not Applicable	This can be used to add custom label(s) to nudr-nrf-client deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to nudr-nrf-client specific deployment.	null	Not Applicable **Note:** ASM related annotations to be added under ASM Specific Configuration section	This can be used to add custom annotation(s) to nudr-nrf-client deployment.

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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	flag for enabling or disabling nudr-config service	true	true/false	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
logging.level.root	Log Level	WARN	Possible Values - WARN INFO DEBUG	Log level of the nudr-config pod
service.http2enabled	Enabled HTTP2 support flag for rest server	true	true/false	Enable/Disable HTTP2 support for rest server
image.name	Docker Image name	ocudr/ nudr_config	Not applicable	
service.customExtension.labels	Custom Labels that needs to be added to nudr-config specific Service.	null	Not applicable	This can be used to add custom label(s) to nudr-config Service.
service.customExtension.annotations	Custom Annotations that needs to be added to nudr-config specific Services.	null	Not applicable	This can be used to add custom annotation(s) to nudr-config Service.
deployment.customExtension.labels	Custom Labels that needs to be added to nudr-config specific Deployment.	null	Not applicable	This can be used to add custom label(s) to nudr-config Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to nudr-config specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to nudr-config Deployment.
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
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
service.port.management	Management port	9000	Not applicable	The actuator management port to be used for nudr-config service

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
service.port.https	HTTPS port	5002	Not applicable	The https port to be used for nudr-config service
service.port.http	HTTP port	5001	Not applicable	The http port to be used in nudr-config service
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
resources.requests.memory	Memory allotment for nudr-drservice pod	2Gi	Not applicable	The memory to be allocated for nudr-config pod during deployment
resources.limits.memory	Memory allotment limitation	2Gi	Not applicable	
resources.requests.cpu	Cpu Allotment for nudr-drservice pod	2	Not applicable	The cpu to be allocated for nudr-config pod during deployment
resources.limits.cpu	Cpu allotment limitation	2	Not applicable	
image.tag	Tag of Image	1.8.0	Not applicable	
deployment.replicaCount	Replicas of nudr-config pod	1	Not applicable	Number of nudr-config pods to be maintained by replica set created with deployment
minReplicas	Minimum Replicas	1	Not applicable	Minimum number of pods
maxReplicas	Maximum Replicas	1	Not applicable	Maximum number of pods
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	30	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.periodSeconds	Time interval for every readiness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	5	Not Applicable Unit: Seconds	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	40	Not Applicable Unit: Seconds	
livenessProbe.periodSeconds	Time interval for every liveness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	

Following table provides the parameters for **nudr-config-server Micro service**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	Flag to enable/ disable nudr-config-server service	true	true/false	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
global.nfName	It is NF name used to add with config server service name.	nudr	Not applicable	
global.imageServiceDetector	Image Service Detector for config-server init container	ocudr/readiness-detector:1.7.1	Not Applicable	
global.envJaegerAgentHost	Host FQDN for Jaeger agent service for config-server tracing	' '	Not Applicable	
global.envJaegerAgentPort	Port for Connection to Jaeger agent for config-server tracing	6831	Valid Port	
envLoggingLevelApp	Log Level	WARN	Possible Values - WARN INFO DEBUG	Log level of the nudr-config-server pod
replicas	Replicas of nudr-config-server pod	1	Not applicable	Number of nudr-config-server pods to be maintained by replica set created with deployment
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
resources.requests.cpu	Cpu Allotment for nudr-drservice pod	2	Not applicable	The cpu to be allocated for nudr-config-server pod during deployment
resources.requests.memory	Memory allotment for nudr-drservice pod	512Mi	Not applicable	The memory to be allocated for nudr-config-server pod during deployment
resources.limits.cpu	Cpu allotment limitation	2	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.limits.memory	Memory allotment limitation	2Gi	Not applicable	
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	70	Not Applicable Unit: Seconds	
readinessProbe.periodSeconds	Time interval for every readiness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	
readinessProbe.timeoutSeconds	Number of seconds after which the probe times out **Note:** Do not change this default value.	3	Not Applicable	
readinessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed **Note:** Do not change this default value.	1	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes tries failureThreshold times before giving up **Note:** Do not change this default value.	3	Not Applicable	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	60	Not Applicable Unit: Seconds	
livenessProbe.periodSeconds	Time interval for every liveness probe check. **Note:** Do not change this value. If there is any delay in pod coming up and probe is killing the pod then you should consider tuning these parameters.	15	Not Applicable Unit: Seconds	
livenessProbe.timeoutSeconds	Number of seconds after which the probe times out Note: Do not change this default value.	3	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed Note: Do not change this default value.	1	Not Applicable	
livenessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up Note: Do not change this default value.	3	Not Applicable	

Following table provides parameters for **nudr-diameterproxy micro service**.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	To enable service.	true	Not applicable	Used to enable or disable service.
image.name	Docker Image name	ocudr/ nudr_diameterproxy	Not applicable	
image.tag	Tag of Image	1.8.0	Not applicable	
image.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
logging.level.root	Log Level	WARN	Possible Values - WARN INFO DEBUG	The log level of the nudr-diameterproxy server pod
deployment.replicaCount	Replicas of the nudr-diameterproxy pod	2	Not applicable	Number of nudr-config-server pods to be maintained by replica set created with deployment

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
minReplicas	min replicas of nudr-diameterproxy	2	Not applicable	Minimum number of pods
maxReplicas	max replicas of nudr-diameterproxy	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	Possible Values- ClusterIP NodePort LoadBalancer	The Kubernetes service type for exposing UDR deployment Note: Suggested to be set as ClusterIP (default value) always
service.diameter.type	Diameter service type	LoadBalancer	Possible Values- ClusterIP NodePort LoadBalancer	The Kubernetes service type for exposing UDR deploymentdiameter traffic goes via diameter-endpoint, not via ingress-gateway
service.port.http	HTTP port	5001	Not applicable	The HTTP port to be used in nudr-diameterproxy service
service.port.https	HTTPS port	5002	Not applicable	The https port to be used for nudr-diameterproxy service
service.port.management	Management port	9000	Not applicable	The actuator management port to be used for nudr-diameterproxy service
service.port.diameter	Diameter port	6000	Not applicable	The diameter port to be used for nudr-diameterproxy service
resources.requests.cpu	Cpu Allotment for nudr-diameterproxy pod	3	Not applicable	The CPU to be allocated for nudr-diameterproxy pod during deployment

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.memory	Memory allotment for nudr-diameterproxy pod	`4Gi`	Not applicable	The memory to be allocated for nudr-diameterproxy pod during deployment
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	The CPU to be max allocated for nudr-diameterproxy pod
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	The memory to be max allocated for nudr-diameterproxy pod
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
drservice.port.http	HTTP port on which dr service is running	5001	Not Applicable	dr-service port is required in diameterproxy application
drservice.port.https	HTTPS port on which dr service is running	5002	Not Applicable	dr-service port is required in diameterproxy application
diameter.realm	Realm of the diameterproxy microservice	oracle.com	String value	Host realm of diameterproxy
diameter.identity	FQDN of the diameterproxy in diameter messages	nudr.oracle.com	String value	identity of the diameterproxy
diameter.strictParsing	Strict parsing of Diameter AVP and Messages	false	Not Applicable	strict parsing
diameter.IO.threadCount	Number of thread for IO operation	0	0 to 2* CPU	Number of threads to handle IO operations in diameterproxy pod if threadcount is 0 then application choose the threadCount based on pod profile size

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
diameter.IO.queueSize	Queue size for IO	0	2048 to 8192	the count should be the power of 2 if queueSize is 0 then application choose the queueSize based on pod profile size
diameter.messageBuffer.threadCount	Number of threads for process the message	0	0 to 2* CPU	Number of threads to handle meassages in diameterproxy pod if threadcount is 0 then application choose the threadCount based on pod profile size
diameter.peer.setting	Diameter peer setting	reconnectDelay: 3 responseTimeout: 4 connectionTimeOut: 3 watchdogInterval: 6 transport: 'TCP' reconnectLimit: 50	Not Applicable	1. reconnect delay for diameter reonnect (in seconds). 2. total turnaround time for process the diameter messages. (in sec) 3. TCP connection timeout time. (in sec) 4. DWR and DWA messages every number of time (in sec) 5. Transport layer 6. reconnect the number of time if diameter peer is down

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
diameter.peer.nodes	diameter server peer nodes list	- name: 'seagull' responseOnly: false namespace: 'seagull1' host: '10.75.185.158' domain: 'svc.cluster.local' port: 4096 realm: 'seagull1.com' identity: 'seagull1a.seagull1.com'	Not applicable	the diameter server peer node information *it should be yaml list *default values are template , how to add peer nodes.
diameter.peer.clientNodes	diameter client peers	- identity: 'seagull1a.seagull1.com' realm: 'seagull1.com' - identity: 'seagull1.com' realm: 'seagull1.com'	Not applicable	the diameter client node information *it should be yaml list *default values is template, how to add peer nodes.
service.customExtension.labels	Custom Labels that needs to be added to nudr-diameterproxy specific Service.	null	Not applicable	This can be used to add custom label(s) to nudr-diameterproxy Service.
service.customExtension.annotations	Custom Annotations that needs to be added to nudr-diameterproxy specific Services.	null	Not applicable	This can be used to add custom annotation(s) to nudr-diameterproxy Service.
deployment.customExtension.labels	Custom Labels that needs to be added to nudr-diameterproxy specific Deployment.	null	Not applicable	This can be used to add custom label(s) to nudr-diameterproxy Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to nudr-diameterproxy specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to nudr-diameterproxy Deployment.

Parameter	**Description**	**Default value**	**Range or Possible Values (If applicable)**	**Notes**
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubeletNote: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	80	Not Applicable Unit: Seconds	
readinessProbe.periodSeconds	Time interval for every readiness probe check.Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	5	Not Applicable Unit: Seconds	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	80	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbe.periodSeconds	Time interval for every liveness probe check. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	20	Not Applicable Unit: Seconds	

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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
global.type	ocudr-ingressgateway service type	LoadBalancer	Possbile Values- ClusterIP NodePort LoadBalancer	
global.metalLbIpAllocationEnabled	Enable or disable Address Pool for Metallb	true	true/false	
global.metalLbIpAllocationAnnotation	Address Pool for Metallb	metallb.universe.tf/address-pool: signaling	Not applicable	
global.staticNodePortEnabled	If Static node port needs to be set, then set staticNodePortEnabled flag to true and provide value for staticNodePort	false	Not applicable	
global.istioIngressTlsSupport.ingressGateway	Supports clear text traffic from outside of the cluster when enabled to try in case of Service Mesh Enabled.	false	true/false	
image.name	Docker image name	ocudr/ocingress_gateway	Not applicable	
image.tag	Image version tag	1.8.1	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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	1.4.0	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	1.4.0	Not applicable	
updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
service.ssl.tlsVersion	Configuration to take TLS version to be used	TLSv1.2	Valid TLS version	These are service fixed parameters
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	
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	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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	
service.initialAlgorithm	Algorithm to be used ES256 can also be used, but corresponding certificates need to be used.	RSA256	RSA256/ES256	
resources.limits.cpu	Cpu allotment limitation	5	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	
resources.limits.initServiceCpu	Maximum amount of CPU that Kubernetes will allow the ingress-gateway init container to use.	1	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.limits.initServiceMemory	Memory Limit for ingress-gateway init container	1Gi	Not Applicable	
resources.limits.updateServiceCpu	Maximum amount of CPU that Kubernetes will allow the ingress-gateway update container to use.	1	Not Applicable	
resources.limits.updateServiceMemory	Memory Limit for ingress-gateway update container	1Gi	Not Applicable	
resources.requests.cpu	Cpu allotment for ocudr-endpoint pod	5	Not Applicable	
resources.requests.memory	Memory allotment for ocudr-endpoint pod	4Gi	Not Applicable	
resources.requests.initServiceCpu	The amount of CPU that the system guarantees for the ingress-gateway init container, and Kubernetes uses this value to decide on which node to place the pod.		Not Applicable	
resources.requests.initServiceMemory	The amount of memory that the system will guarantee for the ingress-gateway init container, and Kubernetes will use this value to decide on which node to place the pod		Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.updateServiceCpu	The amount of CPU that the system will guarantee for the ingress-gateway update container, and Kubernetes will use this value to decide on which node to place the pod.		Not Applicable	
resources.requests.updateServiceMemory	The amount of memory that the system will guarantee for the ingress-gateway update container, and Kubernetes will use this value to decide on which node to place the pod.		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	
initssl	To Initialize SSL related infrastructure in init/update container	false	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
jaegerTracingEnabled	Enable/Disable Jaeger Tracing	false	true/false	
openTracing.jaeger.udpSender.host	Jaeger agent service FQDN	occne-tracer-jaeger-agent.occne-infra	Valid FQDN	
openTracing.jaeger.udpSender.port	Jaeger agent service UDP port	6831	Valid Port	
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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
	Supported cipher suites for ssl	`- 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`	Not applicable	
oauthValidatorEnabled	OAUTH Configuration	false	Not Applicable	
nfType	NFType of service producer	UDR	Not Applicable	Mandatory when oauthValidatorEna ebled is true
producerScope	Comma-seperated list of services hosted by service producer.	nudr-dr,nudr-group-id-map	Valid service list	Mandatory when oauthValidatorEna ebled is true

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
allowedClockSkewSeconds	Set this value if clock on the parsing NF (producer) is not perfectly in sync with the clock on the NF (consumer) that created the JWT.	0	Unit: Seconds	Mandatory when oauthValidatorEna ebled is true
nrfPublicKeyKubeSecret	Name of the secret which stores the public key(s) of NRF.	oauthsecret	Not Applicable	Mandatory when oauthValidatorEna ebled is true
nrfPublicKeyKubeNamespace	Namespace of the NRF publicKey Secret	ocudr	Not Applicable	Mandatory when oauthValidatorEna ebled is true
validationType	Values can be "strict" or "relaxed"."strict" means that incoming requests without "Authorization"(Access Token) header are rejected."relaxed" means that if incoming request contains "Authorization" header, it is validated. If incoming request does not contain "Authorization" header, validation is ignored.	strict	strict/relaxed	Mandatory when oauthValidatorEna ebled is true
producerPlmnMNC	MNC of service producer	14	Valid MNC	
producerPlmnMCC	MCC of service producer	310	Valid MCC	
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	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
maxConnectionsPerIp	Connections from endpoint to other microServices	10	Not Applicable	
serviceMeshCheck	Load balancing will be handled by Ingress gateway, if true it would be handled by serviceMesh	false	true/false	

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/**` `order: 1` `- id: traffic_mapping_http_prov` `uri: http://{{ .Release.Name }}-nudr-drservice:5001` `path: /nudr-dr-prov/**` `order: 2` `- id: traffic_mapping_http_mgmt` `uri: http://{{ .Release.Name }}-nudr-drservice:5001` `path: /nudr-dr-mgm/**` `order: 3` `- id: traffic_mapping_http_udsf` `uri: http://{{ .Release.Name }}-nudr-`	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes

		drservice:5001
		  path: /nudsf-dr/**
		  order: 4
		- id: traffic_mapping_http_group
		  uri: http://{{ .Release.Name }}-nudr-drservice:5001
		  path: /nudr-group-id-map/**
		  order: 5
		- id: traffic_mapping_http_group_prov
		  uri: http://{{ .Release.Name }}-nudr-drservice:5001
		 path: /nudr-group-id-map-prov/**
		  order: 6
		- id: traffic_mapping_http_slf_group_prov
		  uri: http://{{ .Release.Name }}-nudr-drservice:5001
		 path: /slf-group-prov/**
		  order: 7		

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
service.customExtension.labels	Custom Labels that needs to be added to ingressgateway specific service.	null	Not Applicable	This can be used to add custom label(s) to ingressgateway service.
service.customExtension.annotations	Custom Annotations that needs to be added to ingressgateway specific services.	null	Not Applicable	This can be used to add custom annotation(s) to ingressgateway service.
deployment.customExtension.labels	Custom Labels that needs to be added to ingressgateway specific deployment.	null	Not Applicable	This can be used to add custom label(s) to ingressgateway deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to ingressgateway specific deployment.	null	Not Applicable	This can be used to add custom annotation(s) to ingressgateway deployment.
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	30	Not Applicable Unit: Seconds	
readinessProbe.periodSeconds	Time interval for every readiness probe check. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.timeoutSeconds	Number of seconds after which the probe times out Note: Do not change this default value.	3	Not Applicable	
readinessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed Note: Do not change this default value.	1	Not Applicable	
readinessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up Note: Do not change this default value.	3	Not Applicable	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	30	Not Applicable Unit: Seconds	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbe.periodSeconds	Time interval for every liveness probe check. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	15	Not Applicable Unit: Seconds	
livenessProbe.timeoutSeconds	Number of seconds after which the probe times out Note: Do not change this default value.	3	Not Applicable	
livenessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed Note: Do not change this default value.	1	Not Applicable	
livenessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up Note: Do not change this default value.	3	Not Applicable	

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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
enabled	Configuration flag to enable/disable egress gateway	true	true/false	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
image.name	Docker image name	ocudr/ ocegress_gateway	Not applicable	
image.tag	Image version tag	1.8.1	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	1.4.0	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	1.4.0	Not applicable	
updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits. memory	Memory allotment limitation	4Gi	Not applicable	
resources.limits.initServiceCpu	Maximum amount of CPU that Kubernetes will allow the egress-gateway init container to use.	1	Not applicable	
resources.limits.initServiceMemory	Memory Limit for egress-gateway init container	1Gi	Not applicable	
resources.limits.updateServiceCpu	Maximum amount of CPU that Kubernetes will allow the egress-gateway update container to use.	1	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.limits.updateServiceMemory	Memory Limit for egress-gateway update container	1Gi	Not applicable	
resources.requests.cpu	Cpu allotment for ocudr-egressgateway pod	3	Not applicable	
resources.requests.memory	Memory allotment for ocudr-egressgatewaypod	4Gi	Not applicable	
resources.requests.initServiceCpu	The amount of CPU that the system will guarantee for the egress-gateway init container, and Kubernetes will use this value to decide on which node to place the pod		Not Applicable	
resources.requests.initServiceMemory	The amount of memory that the system will guarantee for the egress-gateway init container, and Kubernetes will use this value to decide on which node to place the pod		Not Applicable	
resources.requests.updateServiceCpu	The amount of CPU that the system will guarantee for the egress-gateway update container, and Kubernetes will use this value to decide on which node to place the pod.		Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
resources.requests.updateServiceMemory	The amount of memory that the system will guarantee for the egress-gateway update container, and Kubernetes will use this value to decide on which node to place the pod.		Not Applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not applicable	
service.ssl.tlsVersion	Configuration to take TLS version to be used	TLSv1.2	Valid TLS version	These are service fixed parameters
service.initialAlgorithm	Algorithm to be used ES256 can also be used, but corresponding certificates need to be used.	RSA256	RSA256/ES256	
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	
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	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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	
minAvailable	Number of pods always running	1	Not Applicable	
minReplicas	Min replicas to scale to maintain an average CPU utilization	1	Not applicable	
maxReplicas	Max replicas to scale to maintain an average CPU utilization	4	Not applicable	
log.level.root	Logs to be shown on ocudr-egressgateway pod	WARN	valid level	
log.level.egress	Logs to be shown on ocudr-egressgateway pod for egress related flows	INFO	valid level	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
log.level.oauth	Logs to be shown on ocudr-egressgateway pod for oauth related flows	INFO	valid level	
fullnameOverride	Name to be used for deployment	ocudr-egressgateway	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	occne-tracer-jaeger-agent.occne-infra	Valid FQDN	
openTracing.jaeger.udpSender.port	Jaeger agent service UDP port	6831	Valid Port	
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.
enableOutgoingHttps	Enabling for sending https requests	false	true or false	
oauthClient.enabled	Enable if oauth is required	false	true or false	Enable based on Oauth configuration
oauthClient.dnsSrvEnabled	DNS SRV Enabled for oAuth	false	true/false	
oauthClient.httpsEnabled	Determine if https support is enabled or not which is a deciding factor for oauth request scheme and search query parameter in dns-srv request	false	true/false	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
oauthClient.virtualFqdn	virtualFqdn value which needs to be populated and sent in the dns-srv query.	localhost:port		Mandatory if oauthClient.dnsSrvEnabled is true
oauthClient.staticNrfList	List of Static NRF's	- localhost:port		Mandatory if oauthClient.enabled is true
oauthClient.nfType	NFType of service consumer.	UDR	Not Applicable	Mandatory if oauthClient.enabled is true
oauthClient.consumerPlmnMNC	MNC of service Consumer.	14	Valid MNC	
oauthClient.consumerPlmnMCC	MCC of service Consumer.	310	Valid MCC	
oauthClient.maxRetry	Maximum number of retry that need to be performed to other NRF Fqdn's in case of failure response from first contacted NRF based on the errorCodeSeries configured.	2	Valid Number	Mandatory if oauthClient.enabled is true
oauthClient.apiPrefix	apiPrefix that needs to be appended in the Oauth request flow.	""	Valid String	Mandatory if oauthClient.enabled is true
oauthClient.errorCodeSeries	Determines the fallback condition to other NRF in case of failure response from currently contacted NRF.	4XX	Valid series	Mandatory if oauthClient.enabled is true and requires different error code series
oauthClient.retryAfter	RetryAfter value in milliseconds that needs to be set for a particular NRF Fqdn, if the error matched the configured errorCodeSeries.	5000	Unit: Milliseconds	Mandatory if oauthClient.enabled is true
maxConcurrentPushedStreams	Jetty client configuration	1000	Valid Number	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
maxRequestsQueuedPerDestination	Jetty client configuration	1024	Valid Number	
maxConnectionsPerIp	Max Connections allowed per Ip	4	Valid Number	
connectionTimeout	Connection timeout in milli seconds	10000	Unit: Milliseconds	
requestTimeout	Request Timeout in milli seconds	1000	Unit: Milliseconds	
jettyIdleTimeout	Jetty Idle Timeout in milli seconds	0	Unit: Milliseconds #(ms,<=0 -> to make timeout infinite)	
k8sServiceCheck	Enable this if loadbalancing is to be done by egress instead of K8s	false	true/false	
service.customExtension.labels	Custom Labels that needs to be added to egressgateway specific Service.	null	Not applicable	This can be used to add custom label(s) to egressgateway Service.
service.customExtension.annotations	Custom Annotations that needs to be added to egressgateway specific Services.	null	Not applicable	This can be used to add custom annotation(s) to egressgateway Service.
deployment.customExtension.labels	Custom Labels that needs to be added to egressgateway specific Deployment.	null	Not applicable	This can be used to add custom label(s) to egressgateway Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to egressgateway specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to egressgateway deployment.

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.initialDelaySeconds	Configurable wait time before performing the first readiness probe by the kubelet Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	30	Not Applicable Unit: Seconds	
readinessProbe.periodSeconds	Time interval for every readiness probe check. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	10	Not Applicable Unit: Seconds	
readinessProbe.timeoutSeconds	Number of seconds after which the probe times out Note: Do not change this default value.	3	Not Applicable	
readinessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed Note: Do not change this default value.	1	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
readinessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes will failureThreshold times before giving up Note: Do not change this default value.	3	Not Applicable	
livenessProbe.initialDelaySeconds	Configurable wait time before performing the first liveness probe by the kubelet. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	30	Not Applicable Unit: Seconds	
livenessProbe.periodSeconds	Time interval for every liveness probe check. Note: Do not change this value. If you see delays in pod coming up and probe is killing the pod then you should consider tuning these parameters.	15	Not Applicable Unit: Seconds	
livenessProbe.timeoutSeconds	Number of seconds after which the probe times out Note: Do not change this default value.	3	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
livenessProbe.successThreshold	Minimum consecutive successes for the probe to be considered successful after having failed Note: Do not change this default value.	1	Not Applicable	
livenessProbe.failureThreshold	When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving upNote: Do not change this default value.	3	Not Applicable	

4

Upgrading an Existing Unified Data Repository Deployment

Note:

UDR DOES NOT SUPPORT UPGRADE FROM UDR 1.7.0 OR UDR 1.7.1 TO UDR 1.8.0, WITH/WITHOUT ASM.

To upgrade an existing UDR deployment, you must follow the instructions given in the helm upgrade section.

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

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 1.7.0 version's **ocudr-custom-values.yaml** file before changing any configuration.
- Modify the **<ocudr-custom-values-1.8.0.yaml>** file parameters as per site requirement. For more information on updating the **<ocudr-custom-values-1.8.0.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-1.8.0.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 1.7.0.
2. Use the **rollback.py** script to downgrade to 1.7.0 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 1.7.0 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

Troubleshooting Unified Data Repository

In this chapter, you will learn about the known issues that you may encounter while installing or working on Unified Data Repository and the techniques to troubleshoot these issues. It covers:

- Generic Checklist
- Understanding Log Attribute Details
- Verifying UDR Registration with NRF
- Verifying Container Logs
- Verifying OCUDR Micro Services Logs
- Verifying nudr-bulk-import Tool Logs
- Verifying nudr-migration Tool Logs
- Debugging Errors from Egress Gateway
- Debugging Errors from Ingress Gateway
- Debugging Helm Test Issues
- Debugging HPA Issues
- Debugging HTTPS Support related Issues
- Debugging Notification Issues
- Debugging Pod Creation Failure
- Debugging UDR Registration with NRF Failure
- Debugging UDR with Service Mesh Failure
- Using Logs

Generic Checklist

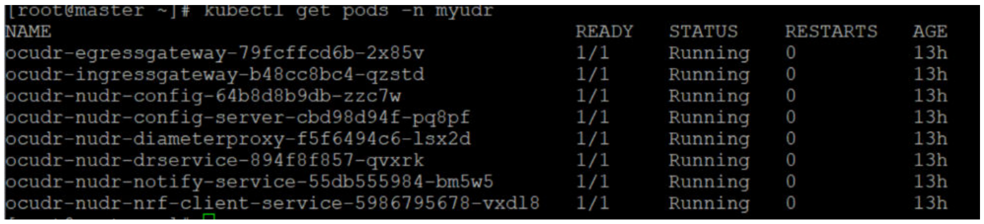
The following generic checklist helps you to ensure that your system is configured properly and there is no issue with basic system setup:

- Execute the following command to **check the installation of kubectl**.
  ```
  $ kubectl
  ```
  If Kubectl is not installed, you can visit https://kubernetes.io/docs/tasks/tools/install-kubectl/
- Execute the following command to **check the installation of helm**.
  ```
  $ helm ls
  ```
  If helm is not installed, execute the following set of commands one after another to install helm:
  1. `curl -o /tmp/helm.tgz https://storage.googleapis.com/kubernetes-helm/helm-v2.9.1-linux-amd64.tar.gz`. Replace with appropriate http link.

2. `tar -xzvf /tmp/helm.tgz -C /usr/local/bin --strip-components=1 linux-amd64/helmrm -f /tmp/helm.tgz`
3. `kubectl create serviceaccount --namespace kube-system tiller`
4. `kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller`
5. `helm init --service-account tiller`
6. `kubectl get po -n kube-system` # Wait for tiller pod to be up
7. `helm ls` # Does not return an error. Try again if returns an error as tiller pod may be coming up.
8. `helm install`. If this command fails immediately with syntax error, check the syntax and values in the values.yaml file. [If values.yaml file is used in helm install command, else contact the UDR development team.]

- Execute the following command to **check the installation of UDR**.
  `$ kubectl get pods -n <ocudr-namespace>`

Figure 5-1 Sample Output: UDR Pods Status

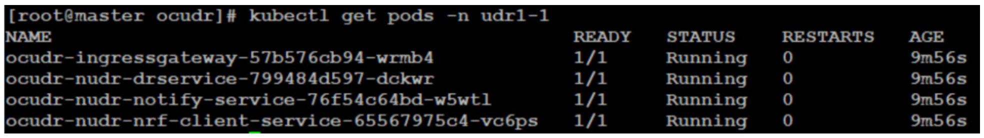


```
[root@master ~]# kubectl get pods -n myudr
NAME                                          READY   STATUS    RESTARTS   AGE
ocudr-egressgateway-79fcffcd6b-2x85v          1/1     Running   0          13h
ocudr-ingressgateway-b48cc8bc4-qzstd          1/1     Running   0          13h
ocudr-nudr-config-64b8d8b9db-zzc7w            1/1     Running   0          13h
ocudr-nudr-config-server-cbd98d94f-pq8pf      1/1     Running   0          13h
ocudr-nudr-diameterproxy-f5f6494c6-lsx2d      1/1     Running   0          13h
ocudr-nudr-drservice-894f8f857-qvxrk          1/1     Running   0          13h
ocudr-nudr-notify-service-55db555984-bm5w5    1/1     Running   0          13h
ocudr-nudr-nrf-client-service-5986795678-vxdl8  1/1   Running   0          13h
```

In the figure given above, the **STATUS** of all the pods is 'Running'.

- Execute the following command **to view all the events** related to a particular namespace.
  `kubectl get events -n <ocudr-namespace>`
- **Verify UDR Pods:** Execute the following command to verify whether UDR specific pods are working as expected:
  `$ kubectl get pods -n <ocudr-namespace>`

Figure 5-2 Sample Output: UDR Pods Status



```
[root@master ocudr]# kubectl get pods -n udr1-1
NAME                                           READY   STATUS    RESTARTS   AGE
ocudr-ingressgateway-57b576cb94-wrmb4          1/1     Running   0          9m56s
ocudr-nudr-drservice-799484d597-dckwr          1/1     Running   0          9m56s
ocudr-nudr-notify-service-76f54c64bd-w5wtl     1/1     Running   0          9m56s
ocudr-nudr-nrf-client-service-65567975c4-vc6ps 1/1     Running   0          9m56s
```

In the figure given above, you can see that the status of all the pods is 'Running'.

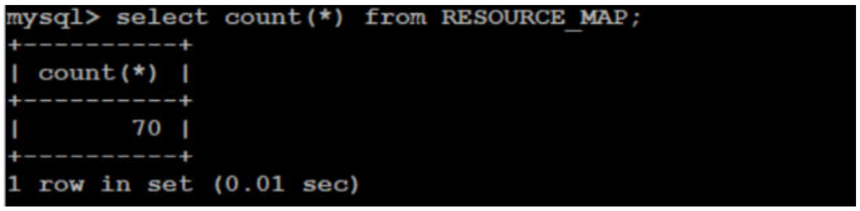
Note:

The number of pods for each service depends on helm configuration. In addition, all pods should be in ready state and you need to ensure that there are no continuous restarts.

- **Verify Database Connectivity:** After verifying UDR pods, login to NDB cluster and verify the creation of udrdb with all the tables. To check the entries in the database tables, you need to execute following command:
  `select count(*) from RESOURCE_MAP`

  It ensures that the connection is fine and the database is created successfully. This count differs based on the **udrServices** option selected under global section in values. But this table cannot be empty.

Figure 5-3 Sample Output: Verifying Table Entries in Database



```
mysql> select count(*) from RESOURCE_MAP;
+-----------+
| count(*)  |
+-----------+
|        70 |
+-----------+
1 row in set (0.01 sec)
```

- **Verify Subscribers:** To verify UDR subscribers, you need to verify the provisioning flow on UDR. You can use the following provisioning URL supported on UDR to verify the provisioning flow:
  - If you use external tools like postman and http2 curl, then follow this URL:
    `http://<ocudr-ingress-gateway-ip>:<http-external-port>/nudr-dr-prov/v1/profile-data/msisdn-1111111113`

    In case of curl, the client should support a http2 curl utility.
  - If https is enabled in UDR ingress gateway, then follow this URL:
    `https://<ocudr-ingress-gateway-ip>:<https-external-port>/nudr-dr-prov/v1/profile-data/msisdn-1111111113`

  Verifying provisioning flow on UDR also confirms udrdb status on the NDB cluster.
- **Verify Logs:** Check the logs of nudr-nrf-client-service for no 503 errors. This helps to find out if all the fqdn configured, as part of helm configurations, in values are resolvable.
- **Verify NRF registration:** Once the deployment has passed the above checks, verify the **udr_nrf_registration_success_total** metric on prometheus after couple of minutes of UDR deployment.

Understanding Log Attribute Details

With the help of following table, you can easily understand the UDR Log attributes (fields) detail.

Log Attribute	Details	Example Value
level	Log Level of log printed	DEBUG
loggerName	Class which printed the log	org.springframework.jdbc.datasource.DataSourceUtils
message	Displays the application generated message as an output	Fetching JDBC Connection from DataSource
instant	Epoch time	{"epochSecond":1599703750, "nanoOfSecond":210064000}
threadId	Displays the ID of the thread that generated the logging event as an output.	23
Timestamp	Timestamp when log was printed	20-09-10 02:09:10.210+0000
Application	NF Application Name	ocudr
Engineering version	Engineering version of software	1.8.0
Marketing version	Marketing version of software	1.8.0.0.0
Microservice	Microservice name	ocudr-nudr-drservice
Namespace	Namespace of udr	ocudr
Cluster	Cluster Name	ocudr
Node	Node Name	5g-udr-dev-1-k8s-node-2
Pod	Pod Name	ocudr-nudr-drservice-5fd845f79d-2jmrn

Verifying UDR Registration with NRF

Execute the following commands to verify whether UDR is registered with NRF.

- **With HTTP1 messaging**
  `curl -v -X GET --url 'http://<FQDN:PORT of NRF-API_Gateway>/nnrf-nfm/v1/nf-instances?nf-type=UDR'`

  Example: `curl -v --http2-prior-knowledge -X GET --url 'http://ocnrf-ingressgateway.ocnrf/nnrf-nfm/v1/nf-instances?nf-type=UDR'`
- **With HTTP2 messaging**
  `curl -v --http2-prior-knowledge -X GET --url 'http://<FQDN:PORT of NRF-API_Gateway>/nnrf-nfm/v1/nf-instances?nf-type=UDR'`

  Example: `curl -v --http2-prior-knowledge -X GET --url 'http://ocnrf-ingressgateway.ocnrf/nnrf-nfm/v1/nf-instances?nf-type=UDR'`

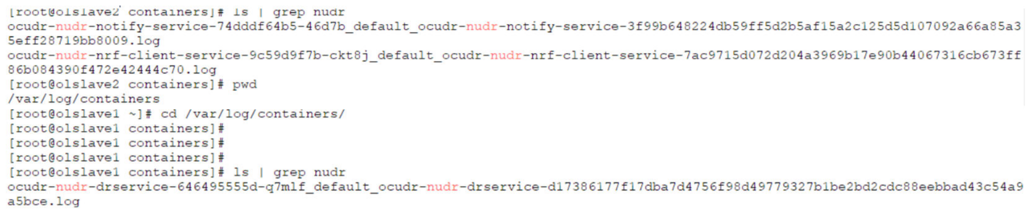
Note:

User should have curl version that supports --http2-prior-knowledge option.

Verifying Container Logs

You can check the container logs in the **/var/log/containers** location on the appropriate nodes where the pods are running.

Figure 5-4 Container Logs



```
[root@olslave2 containers]# ls | grep nudr
ocudr-nudr-notify-service-74dddf64b5-46d7b_default_ocudr-nudr-notify-service-3f99b648224db59ff5d2b5af15a2c125d5d107092a66a85a3
5eff28719bb8009.log
ocudr-nudr-nrf-client-service-9c59d9f7b-ckt8j_default_ocudr-nudr-nrf-client-service-7ac9715d072d204a3969b17e90b44067316cb673ff
86b084390f472e42444c70.log
[root@olslave2 containers]# pwd
/var/log/containers
[root@olslave1 ~]# cd /var/log/containers/
[root@olslave1 containers]#
[root@olslave1 containers]#
[root@olslave1 containers]#
[root@olslave1 containers]# ls | grep nudr
ocudr-nudr-drservice-646495555d-q7mlf_default_ocudr-nudr-drservice-d17386177f17dba7d4756f98d49779327b1be2bd2cdc88eebbad43c54a9
a5bce.log
```

Verifying OCUDR Microservices Logs

In this section, you will learn to check logs of the following microservices:

- OCUDR-NUDR-DRSERVICE
- NRF-CLIENT-SERVICE
- NUDR-NOTIFY-SERVICE
- NUDR-CONFIG-SERVICE
- NUDR-CONFIG-SERVER
- NUDR-DIAMETERPROXY Service
- OCUDR-NUDR-BULK-IMPORT
- NUDR-MIGRATION

Checking Logs in OCUDR-NUDR-DRSERVICE

OCUDR-NUDR-DRSERVICE dumps all the header while processing messages. User should search for "Before Request/After Request" header in the messages.
If nudr-drservice requests are failing, check the count of **udr_schema_operations_failure_total measurement**. If this count is increasing:

- Check the content of incoming requests
- Ensure that the incoming json data blob is proper
- Connectivity between microservices are mysql DB nodes
- Try not to insert duplicate keys
- Ensure DB nodes have enough resources available

To view logs, execute the following command:
`kubectl logs -f <nudr-drservice pod> -n <ocudr-namespace>`

To check logs directly on the pods, execute the following command:

```
kubectl exec -it ocudr-nudr-drservice-779c67b9f-sjcmv bash
```

To change logging level in the ocudr-nudr-drservice using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/upgrade.
2. Change the value of "logging level root" attribute under "ocudr" to "*INFO*".

> **Note:**
>
> OCUDR supports logging level values: DEBUG, INFO, WARN and ERROR.

3. Execute the following helm upgrade command to change the log level:
   `helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with logging level as INFO> --version <helm version>`

Checking Logs in NUDR-NRF-CLIENT-SERVICE

If the count of **udr_nrf_livenessProbe_failure_total measure** increases, you need to ensure that helm charts configuration for "nudr-nrf-client-service" is correct and NRF server is up and running fine.

If nudr-nrf-client-service is not able to register with NRF and
there is a difference between "**udr_nrf_registration_requests_total**" and "**udr_nrf_registration_success_total**", then you need to ensure that helm charts configuration for "nudr-nrf-client-service" are correct.

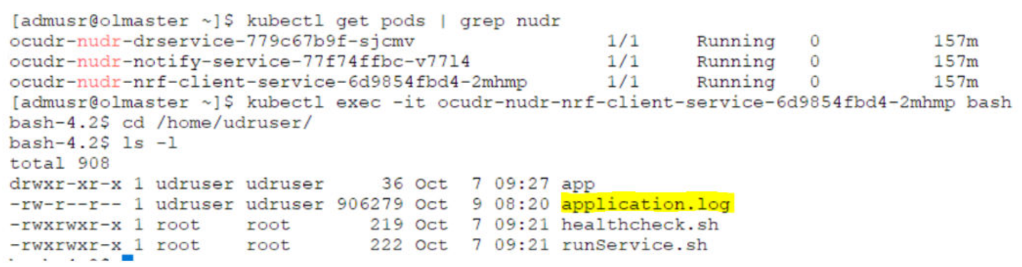
If nudr-nrf-client-service is not able to de-register with NRF and
there is a difference between "udr_nrf_deregistration_requests_total" and "udr_nrf_deregistration_success_total", then you need to ensure that helm charts configuration for "nudr-nrf-client-service" are correct.

To view the NUDR-NRF-CLIENT-SERVICE logs, execute the following command:

```
kubectl logs <nrf-client-pod pod> -n <ocudr-namespace>
```

To check logs directly on the pods, refer to the screen given below:

Figure 5-5   NRF-Client-Service Logs



```
[admusr@olmaster ~]$ kubectl get pods | grep nudr
ocudr-nudr-drservice-779c67b9f-sjcmv                 1/1     Running   0          157m
ocudr-nudr-notify-service-77f74ffbc-v7714            1/1     Running   0          157m
ocudr-nudr-nrf-client-service-6d9854fbd4-2mhmp       1/1     Running   0          157m
[admusr@olmaster ~]$ kubectl exec -it ocudr-nudr-nrf-client-service-6d9854fbd4-2mhmp bash
bash-4.2$ cd /home/udruser/
bash-4.2$ ls -l
total 908
drwxr-xr-x 1 udruser udruser     36 Oct  7 09:27 app
-rw-r--r-- 1 udruser udruser 906279 Oct  9 08:20 application.log
-rwxrwxr-x 1 root    root       219 Oct  7 09:21 healthcheck.sh
-rwxrwxr-x 1 root    root       222 Oct  7 09:21 runService.sh
```

To change logging level in the nrf-client-service using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/upgrade.
2. Change the value of "logging level root" attribute under "nrfclient" to "*INFO*".

> **Note:**
>
> nudr-nrf-client-service supports logging level values: DEBUG, INFO, WARN and ERROR.

3. Execute the following helm upgrade command to change the log level:
   `helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with logging level as INFO> --version <helm version>`

Checking Logs in NUDR-NOTIFY-SERVICE

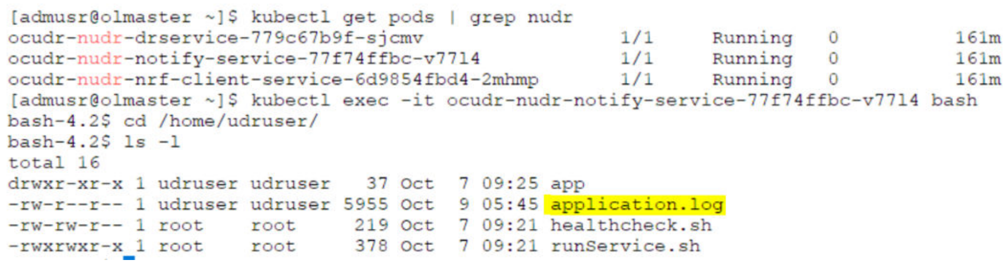
Measurements like **nudr_notif_notifications_ack_2xx_total**, **nudr_notif_notifications_ack_4xx_total**, and **nudr_notif_notifications_ack_5xx_total** gives information about the response code returned in the notification response. If the count of **nudr_notif_notifications_send_fail_total** measurement increases, then you need to ensure that the notification server mentioned in the NOTIFICATION_URI during subscription request is up and running.

To view the NUDR-NOTIFY-SERVICE logs, execute the following command:

```
kubectl logs <nudr-notify-service pod> -n <ocudr-namespace>
```

To check logs directly on the pods, refer to the screen given below:

Figure 5-6 NUDR-NOTIFY-SERVICE Logs



```
[admusr@olmaster ~]$ kubectl get pods | grep nudr
ocudr-nudr-drservice-779c67b9f-sjcmv                1/1     Running   0          161m
ocudr-nudr-notify-service-77f74ffbc-v7714           1/1     Running   0          161m
ocudr-nudr-nrf-client-service-6d9854fbd4-2mhmp      1/1     Running   0          161m
[admusr@olmaster ~]$ kubectl exec -it ocudr-nudr-notify-service-77f74ffbc-v7714 bash
bash-4.2$ cd /home/udruser/
bash-4.2$ ls -l
total 16
drwxr-xr-x 1 udruser udruser   37 Oct  7 09:25 app
-rw-r--r-- 1 udruser udruser 5955 Oct  9 05:45 application.log
-rw-rw-r-- 1 root    root     219 Oct  7 09:21 healthcheck.sh
-rwxrwxr-x 1 root    root     378 Oct  7 09:21 runService.sh
```

To change logging level in the nudr-notify-service using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/ upgrade.
2. Change the value of "logging level root" attribute under "ocudr" to "*INFO*".

> **Note:**
>
> nudr-notify-service supports logging level values: DEBUG, INFO, WARN and ERROR.

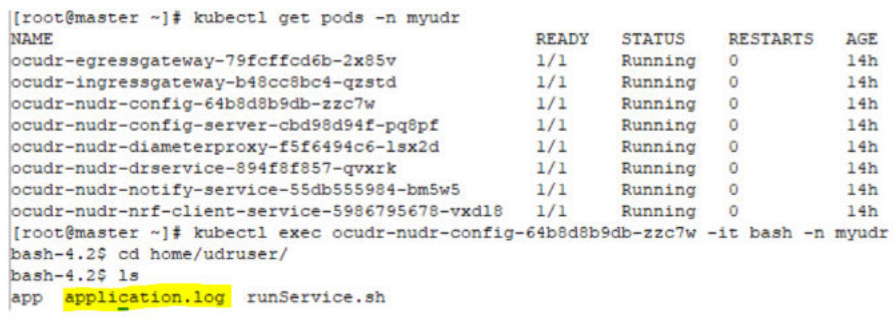
3. Execute the following helm upgrade command to change the log level:
   `helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with logging level as INFO> --version <helm version>`

Checking Logs in NUDR-CONFIG-SERVICE

To view logs, execute the following command:
`kubectl logs <nudr-config pod> -n <ocudr-namespace>`

To check logs directly on the pods, refer to the screen given below:

Figure 5-7 NUDR-CONFIG-SERVICE Logs



```
[root@master ~]# kubectl get pods -n myudr
NAME                                           READY   STATUS    RESTARTS   AGE
ocudr-egressgateway-79fcffcd6b-2x85v           1/1     Running   0          14h
ocudr-ingressgateway-b48cc8bc4-qzstd           1/1     Running   0          14h
ocudr-nudr-config-64b8d8b9db-zzc7w             1/1     Running   0          14h
ocudr-nudr-config-server-cbd98d94f-pq8pf       1/1     Running   0          14h
ocudr-nudr-diameterproxy-f5f6494c6-lsx2d       1/1     Running   0          14h
ocudr-nudr-drservice-894f8f857-qvxrk           1/1     Running   0          14h
ocudr-nudr-notify-service-55db555984-bm5w5     1/1     Running   0          14h
ocudr-nudr-nrf-client-service-5986795678-vxdl8 1/1     Running   0          14h
[root@master ~]# kubectl exec ocudr-nudr-config-64b8d8b9db-zzc7w -it bash -n myudr
bash-4.2$ cd home/udruser/
bash-4.2$ ls
app  application.log  runService.sh
```

To change logging level in the ocudr-nudr-config service using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/ upgrade.
2. Change the value of "logging level root" attribute under "ocudr" to "*INFO*".

   > **Note:**
   >
   > OCUDR supports logging level values: DEBUG, INFO, WARN and ERROR.

3. Execute the following helm upgrade command to change the log level:
   `helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with logging level as INFO> --version <helm version>`

Checking Logs in NUDR-CONFIG-SERVER

To view logs, execute the following command:
`kubectl logs <nudr-config-server pod> -n <ocudr-namespace>`

To change logging level in the ocudr-nudr-config-server service using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/ upgrade.
2. Change the value of "logging level root" attribute under "ocudr" to "*INFO*".

   > **Note:**
   >
   > OCUDR supports logging level values: DEBUG, INFO, WARN and ERROR.

3. Execute the following helm upgrade command to change the log level:
   ```
   helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with
   logging level as INFO> --version <helm version>
   ```

Checking Logs in NUDR-DIAMETERPROXY Service

Debug errors from **ocudr-nudr-diameterproxy**:

- If diameterproxy rejects any request or you are not able to send any request from seagull machines, it means the dictionary file is not loaded correctly to the application. You need to check the dictionary path and change it, if required and redeploy the diameterproxy service. (The dictionary file path should be "`/home/udruser/app/diameter`").
- If diameterproxy answers CEA message with **DIAMETER_UNKNOWN_PEER**, it means client peer is not configured correctly. To resolve this, configure client peer of nudr-diameterproxy service.
- If diameterproxy answers CEA message success and other SH message response as **DIAMETER_UNABLE_TO_COMPLY**, it means the dr-service pod is not up and running or sent sh message is invalid. You can check dr-service failure using **nudr_diameterproxy_rest_failure_res_msgs_total** metrics name and invalid sh message, if **nudr_diameterproxy_total_requests_total** metric is not increasing .
- If there are many error logs in diameterproxy micro service stating connection refused with some IP Address and port, it means specified server peer in helm charts is not running and diameterproxy retries to connect with that peer.
- If you are not getting any PNR messages then check whether dr-service and notify-service is up and running. You need to ensure that server peer configuration is correct.

To view NUDR-DIAMETERPROXY service logs, execute the following command:
```
kubectl logs <nudr-diameterproxy pod> -n <ocudr-namespace>
```

To change logging level in the ocudr-nudr-diameterproxy service using helm:

1. Open the latest ocudr_value.yaml file that is used at the time of ocudr installation/ upgrade.
2. Change the value of "logging level root" attribute under "ocudr" to "*INFO*".

   > **Note:**
   >
   > OCUDR supports logging level values: DEBUG, INFO, WARN and ERROR.

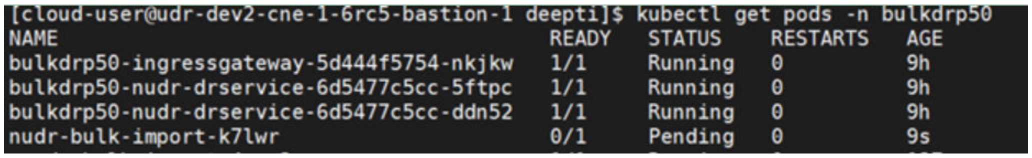
3. Execute the following helm upgrade command to change the log level:
   ```
   helm upgrade ocudr ocudr-helm-repo/ocudr -f <updated values.yaml with
   logging level as INFO> --version <helm version>
   ```

Checking Logs in OCUDR-NUDR-BULK-IMPORT

To check the status of bulk-import pod, execute the following command:

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

Figure 5-8 Checking Status of Bulk Import Pod



```
[cloud-user@udr-dev2-cne-1-6rc5-bastion-1 deepti]$ kubectl get pods -n bulkdrp50
NAME                                        READY   STATUS    RESTARTS   AGE
bulkdrp50-ingressgateway-5d444f5754-nkjkw   1/1     Running   0          9h
bulkdrp50-nudr-drservice-6d5477c5cc-5ftpc   1/1     Running   0          9h
bulkdrp50-nudr-drservice-6d5477c5cc-ddn52   1/1     Running   0          9h
nudr-bulk-import-k7lwr                      0/1     Pending   0          9s
```

Check the logs using kubectl describe command as `kubectl describe pod <bulk-import-pod> -n <namespace>`.

```
Events:
  Type     Reason            Age        From               Message
  ----     ------            ----       ----               -------
  Warning  FailedScheduling  <unknown>  default-scheduler
persistentvolumeclaim "importpersistentclaim" not found
```

- If the logs show the above snippet, you need to ensure that the pvc name in the <pvc yaml file> is same as the claimName under volumes section of the template yaml file.
- If the bulk-import pod is running and you find "**dr-service is down. Job cannot be executed**" in the logs, then you need to ensure that the dr-service and ingressgateway are in running state.
- If the count of measurement of metric, "**nudr_bulk_import_csvfile_records_read_total{Method="DELETE/PUT/POST" , Status="Failure"}**" is increasing , it means the records in the csv file are not valid. You need to provide correct keyType, KeyValue, operationType, nfType and jsonPayload values.
- If you observe the measurement count for the metrics: **nudr_bulk_import_records_processed_total{Method = "POST/PUT/DELETE" , StatusCode="201/204", Status="Success"}** is increasing, it means UDR is processsing the records correctly.
- By measuring the count of the metric, "**Nudr_bulk_import_PCF_total{StatusCode="204/201" ,Status="Success"}**", you can find the number of requests processed successfully for PCF.

Checking Logs in NUDR-MIGRATION

- Using kubectl describe command, you can find details about pods. If a pod is in pending state, it means resources are not present in the CNE and if the pod is in the **ImagePullBackoff** state, it means the image is not able to fetch from repository.
  `kubectl describe pod <pod-name> -n <namespace>`
- If the pod is in running state and still migration did not happen, then:
  - Check the logs and search for ERROR in logs
  - Either the source UDR or target UDR is down. This you can verify in logs.
- If you are not able to connect to 4G UDR, check in logs as **DIAMETER_UNABLE_TO_COMPLY** in CER/CEA messages.
- Check if there is any **UDR/UDA** messages from 4G UDR.

- Check whether **K8S_HOST_IP** port is same as external IP of k8 node that you gave in affinity. If it is not same, it gives **DIAMETER_UNABLE_TO_COMPLY** in **CEA** as response.
- The **INFO** log report displays keys related to any failed record while processing 4G UDR to 5G UDR.
- If you are getting **404** error from 5G UDR, it means 5G UDR is active but records are not available.

Note:

You can use kibana also to view logs.

Verifying nudr-bulk-import Tool Logs

To view the nudr-bulk-import tool logs, execute the following command:

```
kubectl logs <nudr-bulk-import pod> -n <ocudr-namespace>
```

To change the logging level of the nudr-bulk-import tool (using template yaml file - recommended):

1. Open the template yaml file used during nudr-bulk-import tool. Refer #unique_51 for more details.
2. Change the value of **LOGGING_LEVEL_ROOT** attribute available under "*env*" to any one of the following levels:
   - DEBUG
   - INFO
   - WARN
   - ERROR

   An extract from **bulk_import_tool.yaml** is given below:

```
apiVersion: batch/v1
...
...

containers:

-env:

...

- name: LOGGING_LEVEL_ROOT

value: "INFO"
```

Verifying nudr-migration Tool Logs

To view the nudr-migration tool logs, execute the following command:

```
kubectl logs <nudr-migration pod> -n <ocudr-namespace>
```

To change the logging level of the nudr-migration tool (using template yaml file - recommended):

1. Open the template yaml file used during nudr-migration tool. Refer to #unique_52 for more details.
2. Change the value of **LOGGING_LEVEL_ROOT** attribute under "*env*" to any one of the following levels:
   - DEBUG
   - INFO
   - WARN
   - ERROR

   An extract from **nudr_migration.yaml** file is given below:

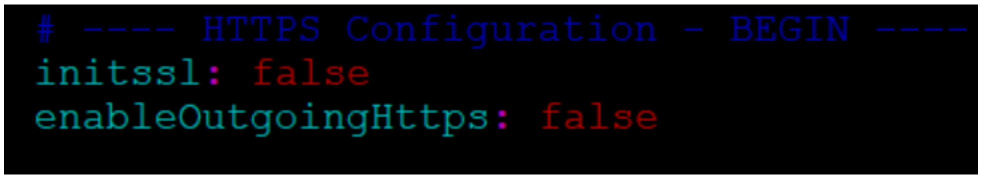
   ```
   apiVersion: batch/v1
   ...
   ...

   containers:

   -env:

   ...

   - name: LOGGING_LEVEL_ROOT

   value: "INFO"
   ```

Debugging Errors from Egress Gateway

If the traffic is not routed via Egress Gateway, you need to check the following:

- Check whether Egress Gateway is enabled or not from global values file.
- Check whether Egress pod is running from kubectl. To check, execute the following command:
  `kubectl get pods -n <Release.name>`
- To enable the outgoing traffic using HTTPS, you need to make the following configuration as true:

  Figure 5-9 Enabling Egress Traffic using HTTPS

  ```
  # ---- HTTPS Configuration - BEGIN ----
  initssl: false
  enableOutgoingHttps: false
  ```

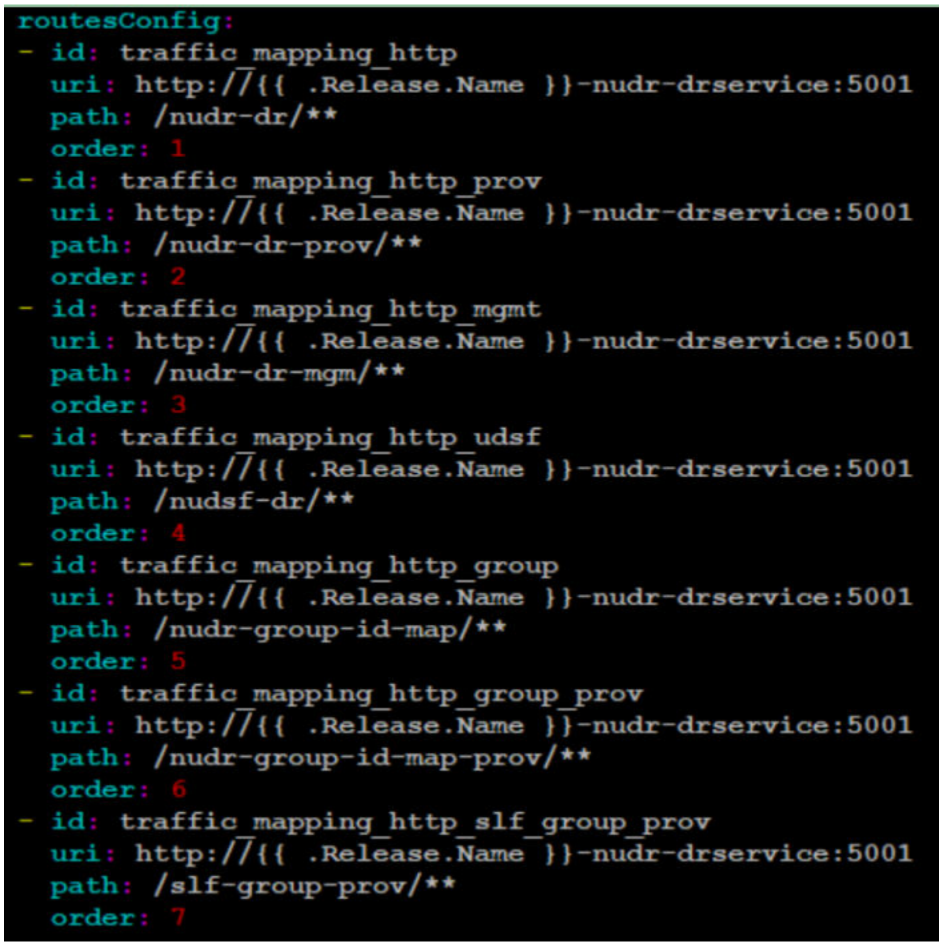
- Create certs and keys uniquely for all Egress and respective Ingress NF's. For more details, check the IngressGateway Container Stuck section in Init State/ Failed. It is same as Ingress debugging.

Debugging Errors from Ingress Gateway

The possible errors that you may encounter from Ingress Gateway are:

- **Check for 500 Error:** If the request fails with 500 status code without Problem Details information, it means that the flow ended in ocudr-ingressgateway pod without route. You can confirm the same in the errors/exception section of the ocudr-ingressgateway pod logs. You also need to check the values.yaml file for the essential route configuration as shown below:

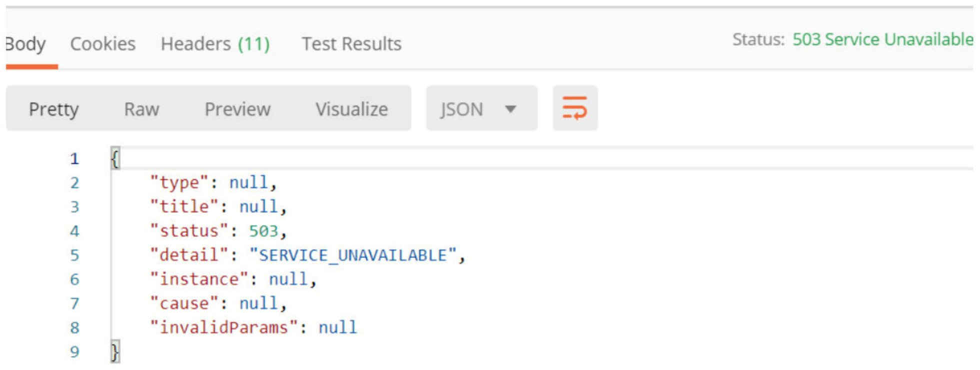
Figure 5-10 Snapshot of Values.yaml file



```
routesConfig:
- id: traffic_mapping_http
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr/**
  order: 1
- id: traffic_mapping_http_prov
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr-prov/**
  order: 2
- id: traffic_mapping_http_mgmt
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-dr-mgm/**
  order: 3
- id: traffic_mapping_http_udsf
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudsf-dr/**
  order: 4
- id: traffic_mapping_http_group
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-group-id-map/**
  order: 5
- id: traffic_mapping_http_group_prov
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /nudr-group-id-map-prov/**
  order: 6
- id: traffic_mapping_http_slf_group_prov
  uri: http://{{ .Release.Name }}-nudr-drservice:5001
  path: /slf-group-prov/**
  order: 7
```

- **Check for 503 Error:** If the request fails with 503 status code with "SERVICE_UNAVAILABLE" in Problem Details, then it means that the nudr-drservice pod is not reachable due to some reason.

Figure 5-11   503 Error Code



```
{
    "type": null,
    "title": null,
    "status": 503,
    "detail": "SERVICE_UNAVAILABLE",
    "instance": null,
    "cause": null,
    "invalidParams": null
}
```

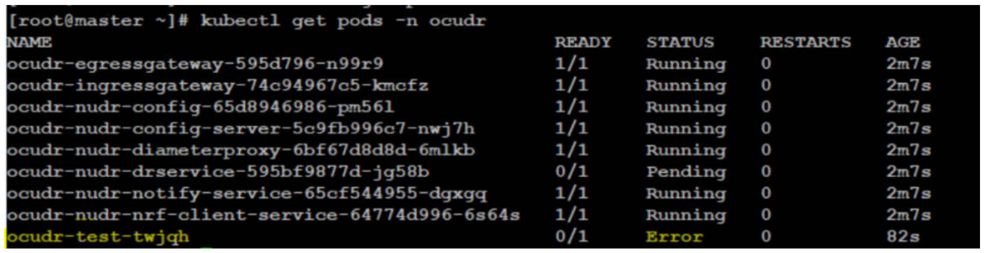
You can confirm the same in the errors/exception logs of the ocudr-ingressgateway pod. Check for ocudr-nudr-drservice pod status and fix the issue.

Debugging Helm Test Issues

To debug Helm Test issues:

- Execute the following command to get the Helm Test pod name.
  `kubectl get pods -n <deployment-namespace>`
- Check for the Helm Test pod that is in error state.

Figure 5-12   Helm Test Pod

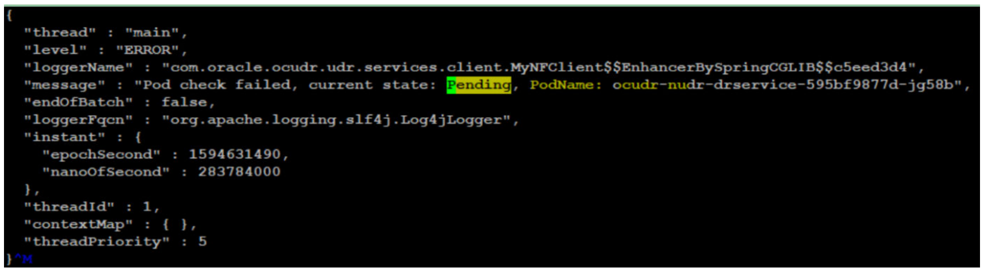


```
[root@master ~]# kubectl get pods -n ocudr
NAME                                                READY   STATUS    RESTARTS   AGE
ocudr-egressgateway-595d796-n99r9                   1/1     Running   0          2m7s
ocudr-ingressgateway-74c94967c5-kmcfz               1/1     Running   0          2m7s
ocudr-nudr-config-65d8946986-pm561                  1/1     Running   0          2m7s
ocudr-nudr-config-server-5c9fb996c7-nwj7h           1/1     Running   0          2m7s
ocudr-nudr-diameterproxy-6bf67d8d8d-6mlkb           1/1     Running   0          2m7s
ocudr-nudr-drservice-595bf9877d-jg58b               0/1     Pending   0          2m7s
ocudr-nudr-notify-service-65cf544955-dgxgq          1/1     Running   0          2m7s
ocudr-nudr-nrf-client-service-64774d996-6s64s       1/1     Running   0          2m7s
ocudr-test-twjqh                                    0/1     Error     0          82s
```

- Execute the following command to check the Helm Test pod:
  `kubectl logs <helm_test_pod_name> -n <deployment_namespace>`

  In the logs, concentrate on ERROR and WARN level logs. There can be multiple reasons for failure. Some of hem are shown below:

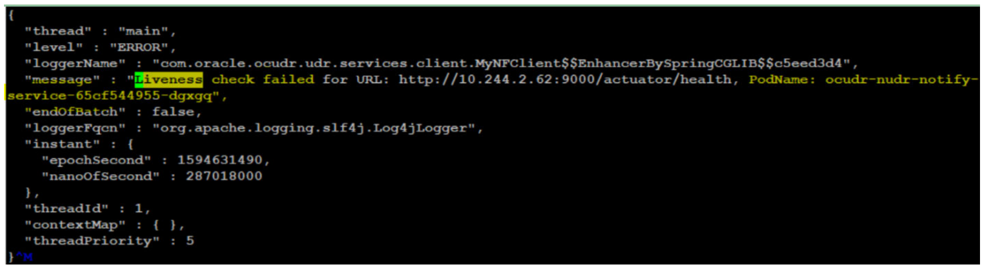
Figure 5-13   Helm Test in Pending State



```
{
  "thread" : "main",
  "level" : "ERROR",
  "loggerName" : "com.oracle.ocudr.udr.services.client.MyNFClient$$EnhancerBySpringCGLIB$$c5eed3d4",
  "message" : "Pod check failed, current state: Pending, PodName: ocudr-nudr-drservice-595bf9877d-jg58b",
  "endOfBatch" : false,
  "loggerFqcn" : "org.apache.logging.slf4j.Log4jLogger",
  "instant" : {
    "epochSecond" : 1594631490,
    "nanoOfSecond" : 283784000
  },
  "threadId" : 1,
  "contextMap" : { },
  "threadPriority" : 5
}^M
```

In this case, check for CPU and Memory availabality in the kubernetes cluster.

Figure 5-14 Pod Readiness Failed



```
{
  "thread" : "main",
  "level" : "ERROR",
  "loggerName" : "com.oracle.ocudr.udr.services.client.MyNFClient$$EnhancerBySpringCGLIB$$c5eed3d4",
  "message" : "Liveness check failed for URL: http://10.244.2.62:9000/actuator/health, PodName: ocudr-nudr-notify-
service-65cf544955-dgxgq",
  "endOfBatch" : false,
  "loggerFqcn" : "org.apache.logging.slf4j.Log4jLogger",
  "instant" : {
    "epochSecond" : 1594631490,
    "nanoOfSecond" : 287018000
  },
  "threadId" : 1,
  "contextMap" : { },
  "threadPriority" : 5
}^M
```

In this case, check for readiness proble url correctness in the particular microservice helm charts under charts folder. In the above case, check for charts of notify service [OR] check if the pod is crashing for some reason when the url configured for readiness probe is correct.

- There are few other cases where the **httpGet** parameter is not configured for Readiness probe. In this case, Helm Test is considered as success for that pod. And if the Pod/PVC list is fetched based on namespace and labelSelector is empty, the helm test is considered as success.

Debugging HPA Issues

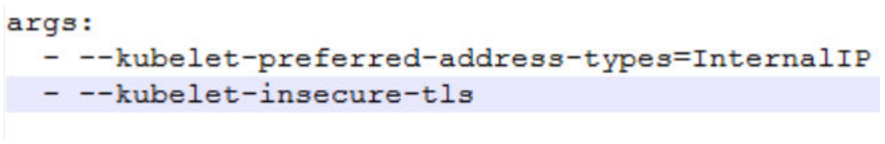
There can be scenarios where HPA running on `nudr-drservice` deployment and `nudr_notify_service` might not get the CPU metrics successfully from the pods. Execute the following command to view the HPA details:

```
kubectl get hpa
```

In this scenario, you need to check the following:

- Check whether metrics server is running on the kubernetes cluster. If it is running, even then the CPU usage pod might not be accessible. In this case, you need to check the metrics-server values yaml file for the args passed as shown below:

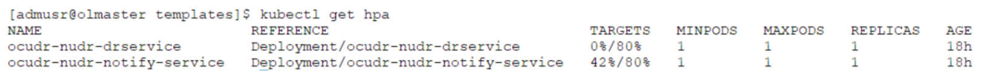
Figure 5-15 metrics-server yaml file



```
args:
  - --kubelet-preferred-address-types=InternalIP
  - --kubelet-insecure-tls
```

- If it requires any update, then do the same and restart the metrics server pod. You have to wait for couple of minutes after starting the metrics server to see the CPU usage update. For this, execute the `kubectl get hpa` command.

Figure 5-16 CPU Usage Update



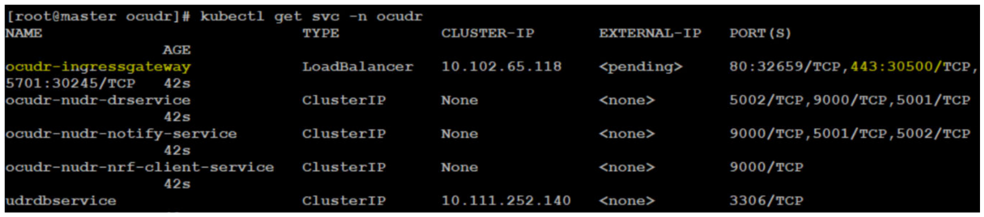
```
[admusr@olmaster templates]$ kubectl get hpa
NAME                         REFERENCE                                 TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
ocudr-nudr-drservice         Deployment/ocudr-nudr-drservice           0%/80%    1         1         1          18h
ocudr-nudr-notify-service    Deployment/ocudr-nudr-notify-service      42%/80%   1         1         1          18h
```

Debugging HTTPS Support related Issues

UDR supports HTTPS and its validations are done at Ingress Gateway of UDR. You may encounter issues related to HTTPS when:

- **HTTPS port is not exposed:** Execute the following command to figure out whether HTTPS port is exposed or not:
  `kubectl get svc --n <ocudr-namespace>`

Figure 5-17 HTTPS Port Exposed

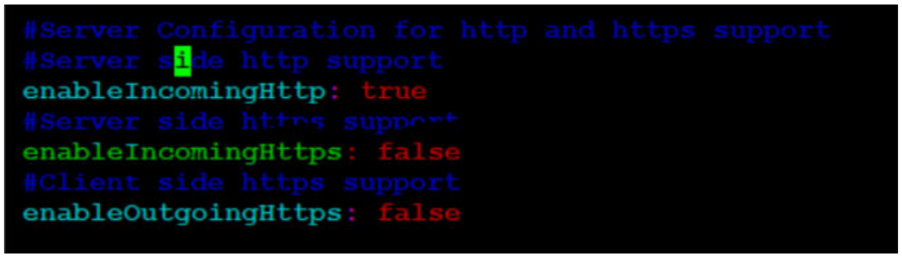


```
[root@master ocudr]# kubectl get svc -n ocudr
NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
                 AGE
ocudr-ingressgateway            LoadBalancer   10.102.65.118    <pending>     80:32659/TCP,443:30500/TCP,
5701:30245/TCP   42s
ocudr-nudr-drservice            ClusterIP      None             <none>        5002/TCP,9000/TCP,5001/TCP
                 42s
ocudr-nudr-notify-service       ClusterIP      None             <none>        9000/TCP,5001/TCP,5002/TCP
                 42s
ocudr-nudr-nrf-client-service   ClusterIP      None             <none>        9000/TCP
                 42s
udrdbservice                    ClusterIP      10.111.252.140   <none>        3306/TCP
```

> **Note:**
>
> In the above screen, the secure port is 443.

If the HTTPS port is not exposed, then enable the configuration information highlighted in the following screen under the ingressgateway section of the values.yaml file.

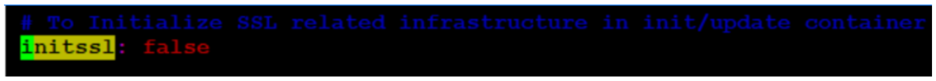
Figure 5-18 Configuration Info under Ingressgateway



```
#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
```

- **IngressGateway Container is stuck in Init State/Failed:**The IngressGateway Container can stuck due to any one of the following reasons:
  - When config initssl is enabled under ingressgateway section of the values.yaml file.

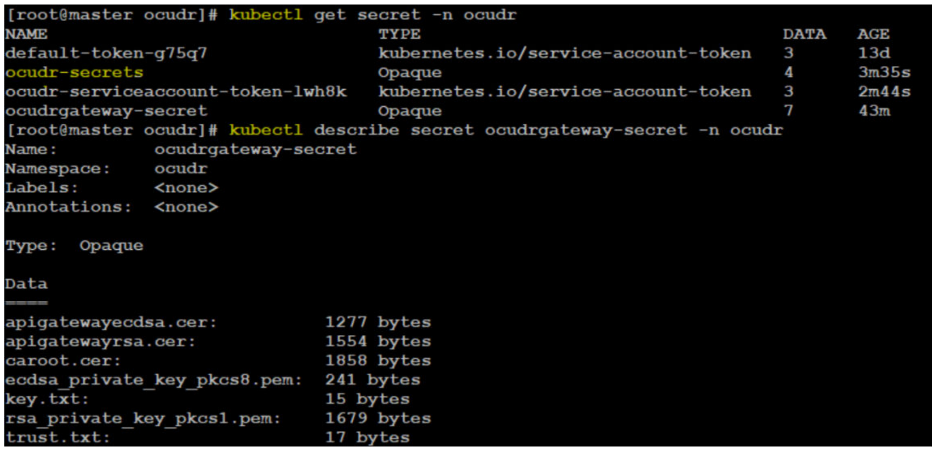
Figure 5-19 config initssl



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

- If config initssl is enabled, then you need to check whether secrets are created with all required certificates. The following screenshot shows the commands that you need to execute to check whether secrets are present and have all the required data.

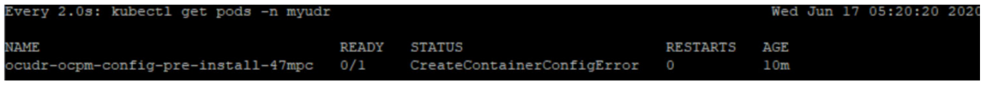
Figure 5-20 Commands to check Secrets



```
[root@master ocudr]# kubectl get secret -n ocudr
NAME                               TYPE                                  DATA   AGE
default-token-g75q7                kubernetes.io/service-account-token   3      13d
ocudr-secrets                      Opaque                                4      3m35s
ocudr-serviceaccount-token-lwh8k   kubernetes.io/service-account-token   3      2m44s
ocudrgateway-secret                Opaque                                7      43m
[root@master ocudr]# kubectl describe secret ocudrgateway-secret -n ocudr
Name:         ocudrgateway-secret
Namespace:    ocudr
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
apigatewayecdsa.cer:          1277 bytes
apigatewayrsa.cer:            1554 bytes
caroot.cer:                   1858 bytes
ecdsa_private_key_pkcs8.pem:  241 bytes
key.txt:                      15 bytes
rsa_private_key_pkcs1.pem:    1679 bytes
trust.txt:                    17 bytes
```

- **Config-Server Container Stuck in Hooks Init State:** The UDR installation stucks in Hooks Init state when there is database connection failure.

Figure 5-21 Config Server Container Status



```
Every 2.0s: kubectl get pods -n myudr                                          Wed Jun 17 05:20:20 2020

NAME                                 READY   STATUS                       RESTARTS   AGE
ocudr-ocpm-config-pre-install-47mpc  0/1     CreateContainerConfigError   0          10m
```

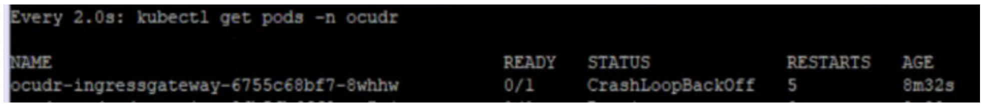
In this case, you need to execute the describe pod command (on the above pod). In most of the cases, it is due to secret not found.

Also, verify the configuration given below to ensure config-server deployment refers to the correct secret values.

```
global:

  dbCredSecretName: 'ocudr-secrets'
```

- **Oauth2 Related Issues:**
If you do not mention the oauth secret name and namespace properly [OR] if the public key in secret is not in correct format, then the Ingress Gateway crashes.

Figure 5-22 Ingress Gateway Crashed



```
Every 2.0s: kubectl get pods -n ocudr

NAME                                  READY   STATUS             RESTARTS   AGE
ocudr-ingressgateway-6755c68bf7-8whhw 0/1     CrashLoopBackOff   5          8m32s
```

Other scenarios are:

- **The secret name in which public key is stored is incorrect:** In this scenario, it is advisable to check the logs of a pod that states "cannot retrieve secret from api server".
- **The public key stored in secret is not in proper format:** The public key format is `{nrfInstanceId}_RS256.pem` (`6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c_RS256.pem`). If the public key is not stored in this format then you need to check the logs of pod that states "Malformed entry in NRF PublicKey Secret with key ecdsa.pem". Here, ecdsa.pem is the public key in oauthsecret.

By using public key in required format, you can resolve these issues. You need to correct the fields with proper secret name and namespace.

Debugging Notification Issues

If UDR does not generate any notification, check the notify service port configuration in the values.yaml file. These ports should be same as ports on which notify service is running.

```
nudr-drservice:
...
...
...
    notify:
        port:
            http: 5001
            https: 5002
```

Debugging Pod Creation Failure

A pod creation can fail due to various reasons. Some of the possible scenarios are explained below:

- **Verifying pod image correctness:** To verify pod image correctness:
  - Verify whether any of the pod is in ImagePullBackOff state.
  - To check whether the image name used for any pod is not correct, verify the values given below in the values.yaml file.

```
global:
  dockerRegistry: ocudr-registry.us.oracle.com:5000
nudr-drservice:
  image:
    name: ocudr/nudr_datarepository_service
    tag: 1.8.0
nudr-nrf-client-service:
  image:
    name: ocudr/nrf_client_service
    tag: 1.8.0


nudr-notify-service:
  image:
    name: ocudr/nudr_notify_service
    tag: 1.8.0

nudr-config:

  image:
    name: ocudr/nudr_config
    tag: 1.8.0

ingressgateway:

  image:
    name: ocudr/ocingress_gateway
    tag: 1.8.0

  initContainersImage:
    name: ocudr/configurationinit
    tag: 1.2.0

  updateContainersImage:
    name: ocudr/configurationupdate
    tag: 1.2.0

egressgateway:

  image:
    name: ocudr/ocegress_gateway
    tag: 1.8.0

  initContainersImage:
    name: ocudr/configurationinit
    tag: 1.2.0

  updateContainersImage:
    name: ocudr/configurationupdate
    tag: 1.2.0

nudr-diameterproxy

  image:
    name: ocudr/nudr_diameterproxy
    tag: 1.8.0
```

– After updating the values.yaml file, execute the following command for helm upgrade:
  ```
  helm upgrade <helm chart> [--version <OCUDR version>] --name
  <release> --namespace <ocudr-namespace> -f <ocudr_values.yaml>
  ```
– If the helm install command is stuck for a long time or fails with timeout error, you need to verify whether the pre-install hooks have come up. Verify

whether there exists any **ImagePullBackOff** check for the correctness of below configurations.

```
global:
  dockerRegistry: ocudr-registry.us.oracle.com:5000

  preInstall:
    name: ocudr/nudr-prehook
    tag: 1.8.0
```

After updating these values, you can purge the deployment and install Helm again.

- **Verifying Resource Allocation Failure:** To verify resource allocation failure:
  - Verify whether any of the pod is in Pending state. If it is there, execute the following command:
    `kubectl describe <nudr-drservice pod id> --n <ocudr-namespace>`
  - Verify whether any warning on Insufficient CPU exists in the describe output of the respective pod. If it exists, it means there are insufficient CPU for the pods to start. You have to either fix the hardware issue or reduce the number of CPUs alloted to a pod in the values.yaml file.

```
nudr-drservice:
...
...
...
resources:

  limits:

    cpu: 3

    memory: 4Gi

  requests:

    cpu: 3

    memory: 4Gi



nudr-notify-service:
...
...
...

resources:

  limits:

    cpu: 3

    memory: 4Gi
  requests:

    cpu: 3

    memory: 4Gi



nudr-config:
...
...
...

resources:

  limits:

    cpu: 3

    memory: 4Gi

  requests:

    cpu: 3

    memory: 4Gi



nudr-config-server:
...
...
...

resources:

  limits:

    cpu: 2

    memory: 2Gi

  requests:

    cpu: 2

    memory: 512Mi



ingress-gateway:
...
...
...
```

```
resources:

  limits:

    cpu: 3

    memory: 4Gi

  requests:

    cpu: 3

    memory: 4Gi
```

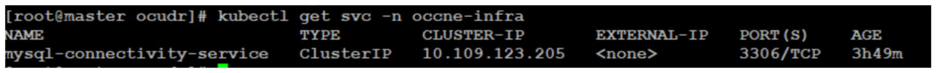
  - After updating the values.yaml file, execute the following command for helm upgrade:
    `helm upgrade <helm chart> [--version <OCUDR version>] --name <release> --namespace <ocudr-namespace> -f <ocudr_values.yaml>`

- **Verifying SQL Exception Failures with nudr-prehook pod:**
  nudr-prehook pod is added as part of 1.7 release. It creates UDR DB along with the tables required. If it does not creates the DB, then to debug the pod failure perform the following steps:
  - Verify whether **helm install** command hangs for longer time or fails with BackOffLimit exceeded error.
  - Watch the **kubectl get pods** command based on the release namespace.
  - Check whether **nudr-preinstall** pod is going to error state. This means the DB creation has failed or connection to DB is not successful.
  - Execute the following command on logs:
    `kubectl logs <nudr-prehook pod id> --n <ocudr-namespace>`
  - Check the log output of the pods for any warning or SQL exceptions using above command continuously. If any warning or SQL execption is found, it means there is an issue with the SQL connection or the SQL Node. Examine each exception thoroughly to find the root caue.
  - Verify the following information in the values.yaml file.

```
global:
...
...
...
mysql:
 dbServiceName: "mysql-connectivity-service.occne-infra" #This
is a read only parameter. Use the default value.
 port: "3306"
```

  - Ensure that the following service is available in the CNE.

Figure 5-23   Service Availability in CNE



```
[root@master ocudr]# kubectl get svc -n occne-infra
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
mysql-connectivity-service    ClusterIP   10.109.123.205   <none>        3306/TCP   3h49m
```

- Check whether kubernetes secrets are present. If secrets exist, then check its encrypted details like username, password and DB name. If these details does not exist, then update the secrets.
- After making any changes, execute the following command to upgrade helm.
  ```
  helm upgrade <helm chart> [--version <OCUDR version>] --name
  <release> --namespace <ocudr-namespace> -f <ocudr_values.yaml>
  ```
  For more details, you can refer to Kubernetes Secret Creation - DBName, Username, Password and Encryption Key.

- **Verifying SQL Exception Failure with nudr-pre-upgrade-hook pod:**
  The **nudr-pre-upgrade-hook** pod takes care of the DB schema upgrade of UDR. It adds new tables if required, along with few more entries to the existing tables. Perform the following steps to debug this pod failure when there is an issue with the DB upgrade:

  1. Checks whether the **helm upgrade** command hangs for long time or fails with BackOffLimit exceeded error.
  2. Ensure that the **pre_upgrade_hook.yaml** file is present in the templates directory of the target charts, with the required annotation. This is for the nudr-pre-upgrade-hook pod to come up.
     ```
     "helm.sh/hook": "pre-upgrade"
     ```
  3. Watch the **kubectl get pods** command based on the release namespace.
  4. Execute the following command on the pods to check if the **nudr-pre-upgrade** pod is going to error state. It means that the DB schema upgrade has failed or connection to DB is not successful.
     ```
     kubectl logs <nudr-pre-upgrade-hook pod id> --n <ocudr-namespace>
     ```
  5. Check the log output of the pod for any warning/SQL Exception. If there is any, it means there is an issue with the SQL connection or the SQL Node. Check the Exception details to get the root cause.
  6. After the upgrade completes, execute the following command to verify whether all the pods are running containers with the updated images.
     ```
     kubectl describe pod <pod id> --n <ocudr-namespace>
     ```

Debugging UDR Registration with NRF Failure

UDR registration with NRF may fail due to various reasons. Some of the possible scenarios are as follows:

- **Verify pod status:** Verify whether all the pods are running or not. Ensure atleast one replica for each microservice is up and running. If it is not running, check for possible reasons. Once the issue resolves, UDR registers successfully with NRF.
- **Verify NRF url correctness:** Execute the following command to check the logs of the ocudr-nudr-nrf-client-service pod:
  ```
  kubectl logs <nrf-client-service pod id> --n <ocudr-namespace>
  ```
  If the logs state that the connection with NRF fails as shown below:

  ```
  10:07:01.335 [scheduling-1] WARN
  ocudr.udr.services.client.RestClient
  - Got error response
  {nfInstanceId=3fd8556a-7804-4abd-8143-640904042d89,
  answerStr=java.net.UnknownHostException: ocnrf-
  ```

```
ingressgateway.mynrf.svc.cluster.local,
 response=<503,java.net.UnknownHostException: ocnrf-
ingressgateway.mynrf.svc.cluster.
local,[]>, nrfBaseUrl=http://ocnrf-
ingressgateway.mynrf.svc.cluster.local/nnrf-nfm/
v1/nf-instances, header={Content-Type=[application/json]},
uri=http://ocnrf-ingressgateway.mynrf.svc.cluster.local/nnrf-nfm/v1/
nf-instances/3fd8556a-7804-4abd-8143-640904042d89}

10:07:01.340 [scheduling-1] WARN
ocudr.udr.services.client.RestClient -
Got error response {answerStr=java.net.UnknownHostException: ocnrf-
ingressgateway.
mynrf.svc.cluster.local, headerMap={Content-Type=[application/
json]}, response=<503,
java.net.UnknownHostException: ocnrf-
ingressgateway.mynrf.svc.cluster.local,[]>,
profile={"nfInstanceId":"3fd8556a-7804-4abd-8143-640904042d89","nfTy
pe":"UDSF",
"nfStatus":"REGISTERED","fqdn":"ocudr-
ingressgateway.myudr.svc.cluster.local",
"udrInfo":{"supiRanges":[{"start":"1000000000
```

Then, verify the baseurl used for NRF in the values.yaml file (as shown below), which is used for connection with NRF.

```
nudr-nrf-client-service:
...
...
...
  host:
  baseurl: "http://ocnrf-ingressgateway.mynrf.svc.cluster.local/
nnrf-nfm/v1/nf-instances"
```

- **Verify UDR fqdn correctness:** Execute the following command to check the logs of the nrf-client-service pod:
  `kubectl logs <nrf-client-service pod id> --n <ocudr-namespace>`

  If the logs state that the FQDN used is not correct then the UDR registration with NRF fails. You need to check the FQDN used in the values.yaml file as follows:

  ```
  nudr-nrf-client-service:
  ...
  ...
  ...
    fqdn: "ocudr-ingressgateway.myudr.svc.cluster.local"
  ```

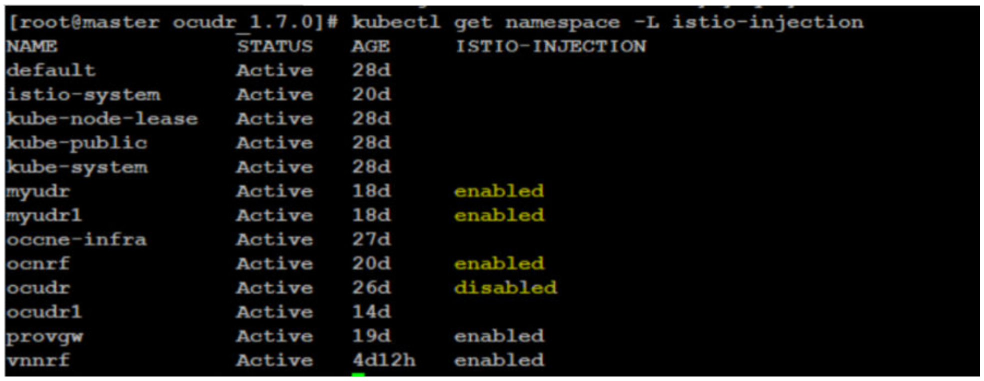
  This helps to connect with NRF.

Debugging UDR with Service Mesh Failure

There are some known failure scenarios that you may encounter while installing UDR with service mesh. The scenarios along with their solutions are as follows:

- **Istio-Proxy side car container not attached to Pod**: This particular failure arise when istio injection is not enabled on the NF installed namespace. Execute the following command to verify the same:
  `kubectl get namespace -L istio-injection`

  Figure 5-24 Verifying Istio-Proxy

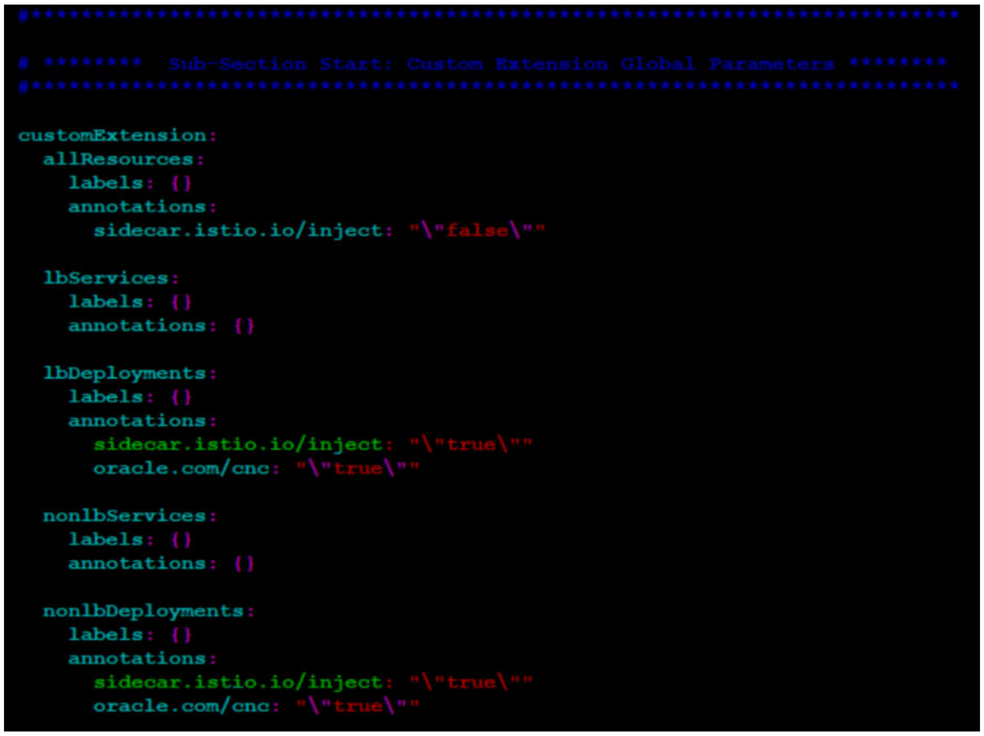
```
[root@master ocudr_1.7.0]# kubectl get namespace -L istio-injection
NAME              STATUS   AGE     ISTIO-INJECTION
default           Active   28d
istio-system      Active   20d
kube-node-lease   Active   28d
kube-public       Active   28d
kube-system       Active   28d
myudr             Active   18d     enabled
myudr1            Active   18d     enabled
occne-infra       Active   27d
ocnrf             Active   20d     enabled
ocudr             Active   26d     disabled
ocudr1            Active   14d
provgw            Active   19d     enabled
vnnrf             Active   4d12h   enabled
```

  To enable the istio injection, execute the following command:

```
kubectl label --overwrite namespace <nf-namespace> istio-
injection=enabled
```

  Other possible reason for this error could be that the below highlighted annotation is missing from the deployment.

Figure 5-25   Global Section - Istio-Proxy Info

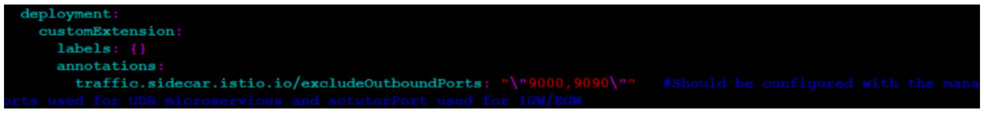


```
#****************************************************************************
# ********  Sub-Section Start: Custom Extension Global Parameters ********
#****************************************************************************

customExtension:
  allResources:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"false\""

  lbServices:
    labels: {}
    annotations: {}

  lbDeployments:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"true\""
      oracle.com/cnc: "\"true\""

  nonlbServices:
    labels: {}
    annotations: {}

  nonlbDeployments:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"true\""
      oracle.com/cnc: "\"true\""
```

You need to add the highlighted annotation as shown above to the global section for **lbDeployments** and **nonlbDeployments** parameters.

- **UDR registration with NRF failed:** This can be due to NF liveness probe failure. You can confirm this on nudr-nrf-client-service pod logs. In this case, you need to ensure that the management port of all UDR microservices are excluded from side car envoy usage. You have to configure proper port as suggested in the below annotation under nudr-nrf-client-service section.

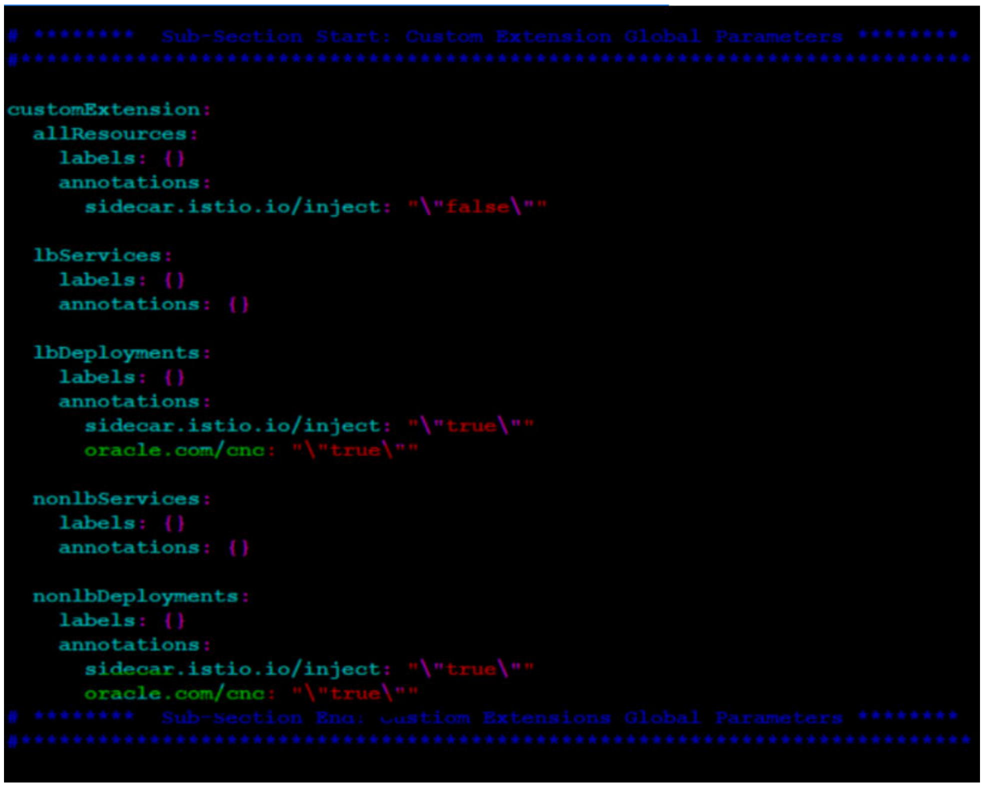
Figure 5-26   Annotation to Configure Port



```
  deployment:
    customExtension:
      labels: {}
      annotations:
        traffic.sidecar.istio.io/excludeOutboundPorts: "\"9000,9090\""   #Should be configured with the mana
orts used for UDR microservices and actutorPort used for IGW/EGW
```

- If there are issues in viewing UDR metrics on OSO prometheus then you have to add the annotation given below to all the deployments for the NF.

Figure 5-27   Annotation to View UDR Metrics



```
# ********  Sub-Section Start: Custom Extension Global Parameters ********
#**************************************************************************

customExtension:
  allResources:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"false\""

  lbServices:
    labels: {}
    annotations: {}

  lbDeployments:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"true\""
      oracle.com/cnc: "\"true\""

  nonlbServices:
    labels: {}
    annotations: {}

  nonlbDeployments:
    labels: {}
    annotations:
      sidecar.istio.io/inject: "\"true\""
      oracle.com/cnc: "\"true\""
# ********  Sub-Section End: Custiom Extensions Global Parameters ********
#**************************************************************************
```

Using Logs

The following table helps you to understand the logs you need to look into, to handle different UDR debugging issues:

SNO	Scenarios	Pod	Logs to be searched	Log Level
1	Registration with NRF Successful	nrf-client-service	Register completed successfully / "nfServiceStatus":"REGISTERED"	INFO
2	Heartbeat message log	nrf-client-service	Update completed successfully	INFO
3	NRF configurations reloading	nrf-client-service	NRF client config reloaded	INFO
4	Check for exiting NF Instance Entry	nrf-client-service	No registered NF instance exists	WARN
5	Started Application	nrf-client-service	Successful application start	INFO
6	Started Application	nudr-drservice	Successful application start	INFO
7	NRF Client Config Initialized	nrf-client-service	Initialize NRF client configuration	INFO
8	FQDN/BASEURL/ livenessProbeUrl Improper	nrf-client-service	response=<503,java.net.UnknownHostException	WARN

SNO	Scenarios	Pod	Logs to be searched	Log Level
9	nudr-drservice liveness probe failure	nrf-client-service	NFService liveness probe failed	WARN
10	SQL Exception during start up	nudr-drservice	java.sql.SQLExceptio n	WARN
11	DB connection pool Established	nudr-drservice	HikariPool-1 - Start completed	INFO
12	Error Code Mapping configurations loaded	nudr-drservice	Loaded Error Code Mapping Configuration	INFO
13	Error Code Mapping configurations loaded	nudr-drservice	Loaded Error Reason Mapping Configuration	INFO
14	Error Code Mapping configurations loaded	nudr-drservice	Loaded Error Title Mapping Configuration	INFO
15	Error Code Mapping configurations loaded	nudr-drservice	Loaded Error Type Mapping Configuration	INFO
16	Check if Ports successfully listening	nudr-drservice	Undertow started on port(s)	INFO
17	Check for message received	nudr-drservice	Before request [uri=<uri-sent excluding ip and port>	DEBUG
18	Check for message processed	nudr-drservice	After request [uri=<uri-sent excluding ip and port>	DEBUG
19	URI Pattern not supported	nudr-drservice	None match pattern found for URL	WARN
20	Check if Ports successfully listening	nrf-client-service	Undertow started on port(s)	INFO
21	Pod exit	nudr-drservice	HikariPool-1 - Shutdown completed	INFO
22	DB username/ DB password invalid	nudr-drservice	Access denied for user	WARN
23	Registration with NRF failed	nrf-client-service	Register failed	ERROR
24	De registration with NRF successful	nrf-client-service	Deregister completed successfully	INFO
25	De registration with NRF failed	nrf-client-service	Deregister failed	ERROR
26	NF Profile update failed	nrf-client-service	Update failed	ERROR

6

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>
```

A

ASM Specific Configuration

To configure ASM, you have to:

- Add the following annotation under **Global section** of UDR deployment.

```
 # ********  Sub-Section Start: Custom Extension Global Parameters
********

#*******************************************************************
*******
global:
  customExtension:
    allResources:
      labels: {}
      annotations:
        sidecar.istio.io/inject: "false"

    lbServices:
      labels: {}
      annotations: {}

    lbDeployments:
      labels: {}
      annotations:
        sidecar.istio.io/inject: "true"
        oracle.com/cnc: "true"
    nonlbServices:
      labels: {}
      annotations: {}

    nonlbDeployments:
      labels: {}
      annotations:
        sidecar.istio.io/inject: "true"
        oracle.com/cnc: "true"

  # ********  Sub-Section End: Custiom Extensions Global Parameters
********

#*******************************************************************
*******
```

- Enable Service Mesh Flag under **ingressgateway section**.

```
ingressgateway:

  global:
    # In case of ASPEN Service Mesh enabled, to support clear text
traffic
from outside of the cluster below flag needs to be true.

    istioIngressTlsSupport:

      ingressGateway: true

  # Mandatory: This flag needs to set it "true" is Service Mesh
would be present
where UDR will be deployed
  serviceMeshCheck: true
```

- Change Ingress Gateway Service Type to ClusterIP under **ingressgateway section**.

```
ingressgateway:
  global:
    # Service Type
    type: ClusterIP
```

- Exclude actuator ports from Aspen Mesh to avoid traffic through side car. These ports are used as actuator ports (used for readiness/liveness checks) for Ingress Gateway and UDR microservices. The default actuator port (service.port.management) used for UDR microservices is 9000 and Ingress/ Egress Gateway is 9090 (ingressgateway.ports.actuatorPort). If there is no change in default ports, you can use the annotation given below.

```
nudr-nrf-client-service:
  deployment:
    customExtension:
      labels: {}
      annotations:
        traffic.sidecar.istio.io/excludeOutboundPorts: "9000,9090"
```

- Create a destination rule and service entry to enable MYSQL connectivity service to establish a connection between UDR/SLF and NDB cluster. This is outside ASM. The sample templates are as follows:
  Creating a Service for External MySQL instance

```
apiVersion: v1
kind: Endpoints
metadata:
  name: mysql-connectivity-service-headless
  namespace: <ocudr-namespace>
subsets:
- addresses:
  - ip: <sql-node1-ip>
  - ip: <sql-node2-ip>
 ports:
  - port: 3306
    protocol: TCP
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: mysql-connectivity-service-headless
  namespace: <ocudr-namespace>
spec:
  clusterIP: None
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-connectivity-service
  namespace: <ocudr-namespace>
spec:
  externalName: mysql-connectivity-service-headless.<ocudr-
namespace>.svc.cluster.local
  sessionAffinity: None
  type: ExternalName
```

Creation of Service Entry and DestinationRule for External DB instance

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: mysql-external-se
  namespace: <ocudr-namespace>
spec:
  hosts:
  - mysql-connectivity-service-headless.<ocudr-
namespace>.svc.cluster.local
  ports:
  - number: 3306
    name: mysql
    protocol: MySQL
  location: MESH_EXTERNAL
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: mysql-external-dr
  namespace: <ocudr-namespace>
spec:
  host: mysql-connectivity-service-headless.<ocudr-
namespace>.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE
```