Oracle® Communications

Unified Data Repository Provisioning Gateway Guide

Release 1.8
F35119-01
September 2020

Oracle Communications Unified Data Repository Provisioning Gateway Guide, Release 1.8

F35119-01

Primary Authors: (primary author), (contributing author), (contributing author), (contributor), (contributor)

Contents

What's New in This Guide

This section shares the list of new features introduced in every Provisioning Gateway release. For more release specific information, please 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.
- Auditor service, which audits the SLF segments and creates a report for descrepencies found in subscriber records.

1 Introducing Provisioning Gateway

In this chapter, you will get an overview of Provisioning Gateway, its architecture and the operations it supports.

Overview

In this section, you will get an overview on Provisioning Gateway.

Oracle Provisioning Gateway is implemented as a cloud-native function. It offers a rest interface for SLF data provisioning. It relays the request received by the provisioning system to multiple 5G UDRs. It then consolidates the response received from each UDR and sends back the final response to the provisioning system.

With the help of Ingress Gateway, Provisioning Gateway provides an HTTP2 based secured RESTful interface for provisioning clients.

It also integrates with Egress API gateway for traffic towards UDR/SLF.

Architecture

In this section, you will learn about Provisioning Gateway architecture.

Provisioning Gateway:

- Interfaces with provisioning system on the northbound and UDR (5G) on the southbound interface.
- Provisioning system uses REST-JSON based interface to send requests to UDR.
- Forks requests to multiple UDRs (multiple segments like UDR1, UDR2). The number of UDRs to which requests are forked is configurable
- Traffic between Provisioning Gateway and UDR is managed via Egress Gateway.
- Can be deployed in multiple setups. Each one is stateless and does not communicate with each other. If one of the Provisioning Gateway fails, provisioning system can send commands to second Provisioning Gateway to continue provisioning.
- Micro services are:
  - **Ingress API Gateway:** receives requests from provisioning system and forwards the same to provisioning gateway service. It also provides TLS for secure request handling and loadbalances among all provisioning gateway service pods.
  - **Egress API Gateway:** handles egress traffic towards UDR/SLF.
  - **Provisioning Gateway Service:** is the core service, which handles the main business logic of forking requests to multiple UDR/SLF. The service ensures the provisioning operations are atomic across segments i.e. returns success only when requests are successful on all UDR/SLFs.

  - **Auditor Service:** audits the subscriber records across segments and creates a report for descrepencies in the subscriber information.
- CNF is integrated with OCCNE services like EFK, Prometheus/Grafana and Jaegar.

List of Operations Supported

In this section, you will learn about the Network Function (NF) related operations for provisioning.

The NF-group-id related operations for provisioning are:

- **Creates/Updates SLF data for a subscriber:** Adds or updates the SLF data related to nf-group-id.
- **Get SLF Data:** Retrieves the nf-group information of a subscriber.
- **Delete SLF Data:** Deletes the nf-group information and related data for a subscriber.

References

In this section, you will learn about Provisioning Gateway references.

Refer to the following documents for more information on Provisioning Gateway usage in 5G cloud native environment.

- Unified Data Repository Installation and Upgrade Guide
- Unified Data Repository User's Guide
- Cloud Native Core Network Function Data Collector User's Guide

2
Installing Provisioning Gateway

In this chapter, you will learn to install Provisioning Gateway.

Planning Your Installation

In this section, you will learn to plan Provisioning Gateway installation.

Pre-installation Tasks:

- Checking the software requirements
- Checking the environment setup

Checking the Software Requirements

In this section, you will learn about softwares required to install provisioning gateway.

Before installing Provisioning Gateway, 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	Chart 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 Provisioning Gateway in any other environment, then the above softwares must be installed before installing Provisioning Gateway.

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
```

Checking the Environment Setup

In this section, you will learn to setup an environment to install Provisioning Gateway.

Before installing Provisioning Gateway, your system should have the following:

- **Provisioning Gateway Software:** The Provisioning Gateway software consists of:
  - **Provisioning Gateway Helm Chart:** reflects the Provisioning Gateway software version. It comes in the form of a zipped tar file.
  - **Software images of the micro-services:** are available in the form of docker images and/or tar file.

  > **Note:**
  >
  > For more details about Provisioning Gateway software, see Checking the Software Requirements
- **Network Access:** The Kubernetes cluster hosts must have network access to:
  - Local docker image repository where the Provisioning Gateway images are available.
  - Local helm repository where the Provisioning Gateway helm charts are available.

  > **Note:**
  >
  > Execute all the kubectl and helm commands used in this document on a system depending on the infrastructure of the deployment. It may be some 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.
  - Helm repository configured on the client.

- Network access to the Kubernetes cluster.
- Necessary environment settings to run the 'kubectl' commands. The environment should have privileges to create a 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.

Installation Sequence

In this section, you will learn about Provisioning Gateway installation sequence:

The installation sequence of Provisioning Gateway is as follows:

1. Installation Preparation
2. **Namespace Creation:** Before creating a namespace, it is important to verify whether any namespace is existing in the system or not. If a namespace does not exist, you must create it. The steps to verify and create a namespace are as follows:
   a. Execute the following command to verify the existence of required namespace in system:
   `kubectl get namespace`
   b. 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 provgw`

> **Note:**
>
> This is an optional step. If the required namespace already exists, proceed with next step.

3. **Service Account, Role and RoleBinding Creation:** A sample command to create the resources is as follows:
   Example: `kubectl -n <provgw-namespace> create -f provgw-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 provgw namespace and provgw helm release name.

```
#
# Sample template start
#
apiVersion: v1
kind: ServiceAccount
```

```
metadata:
  name: <helm-release>-serviceaccount
  namespace: <namespace>
---

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

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

#
# Sample template end
#
```

4. **PVC Creation for persistence storage of audit reports**: To create a PVC:
   a. Create a yaml file with following content:

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

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

> **Note:**
>
> Do not change the **name** under **metadata** used in the above template for auditor service.

You should configure the respective/correct storage class under which you want to create PV.

b. Execute the following command to create a PVC:
`kubectl -f <pvc.yaml> -n <namespace>`

c. Execute the following command to verify PVC creation:
`kubectl get pvc -n <namespace>`

5. **Kubernetes Secret Creation** (provgw-ingress-secret) for storing private keys and certificates for https:

> **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 provgw. This section shares only samples to create them.

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

a. Execute the following command to generate RSA private key:

```
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"
```

b. Execute the following command to convert the private key to .pem format:

```
openssl rsa -in rsa_private_key -outform PEM -out
rsa_private_key_pkcs1.pem -passin
            pass:"keystorepasswd" -passout pass:"keystorepasswd"
```

c. Execute the following command to generate certificate using the private key:

```
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
```

d. Execute the following set of commands to create a root Certificate Authority (CA):

```
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
```

e. Execute the following command to sign the server certificate with root CA private key:

```
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"
```

f. Execute the following command to generate ECDSA private key:

```
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
```

g. Execute the following set of commands to generate certificate using the private key:

```
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"
```

h. Execute the following command to sign the server certificate with root CA private key:

```
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"
```

i. Create a key.txt file by entering any password.

```
Example: echo "keystorepasswd" > key.txt
```

j. Create a trust.txt file by entering any password.

```
Example: echo "truststorepasswd" > trust.txt
```

k. Execute the following set of commands to create a Secret:

```
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>
```

6. **provgw-custom-values-1.8.0.yaml File Configuration**: This includes repository path, primary and secondary node and Provisioning Gateway details configuration. Other configurations may change depending on the deployment.

   > **Note:**
   >
   > For more details, you can refer to Customizing Provisioning Gateway.

7. **ProvGateway Deployment and Verification:** You can deploy Provisioning Gateway either with **HELM repository** or with **HELM tar** in Kubernetes cluster. Before deploying Provisioning Gateway, you need to prepare for its installation. Execute the following command to deploy Provisioning Gateway:

   ```
   helm install <helm chart> [--version <ProvGw version>] --
   name <release> --namespace <k8s namespace> -f <provgw-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>/ provgw.
   - **<ProvGw version>:** is the software version (helm chart version) of the Provisioning Gateway. This is optional. If omitted, the default is 'latest' version available in helm repository.
   - **<release>:** is a user defined name to identify the helm deployment. All pod names, service name, deployment name are prepended by the release name.
   - **<k8s namespace>:** is a user defined name used to identify the kubernetes namespace of the Provisioning Gateway. All the Provisioning Gateway micro services are deployed in this kubernetes namespace.
   - **<provgw-custom-values-1.8.0.yaml>:** is the customized provgw-custom-values-1.8.0.yaml file. For more details, refer to Customizing Provisioning Gateway.

   > **Note:**
   >
   > If helm3 is used, then execute the following command for installation:
   > ```
   > helm install -name <release> --namespace <k8s namespace> -
   > f <provgw-custom-values-1.8.0.yaml> <helm chart> [--version
   > <PROVGW version>]
   > ```

   After Provisioning Gateway deployment, you need to verify whether all the services and pods are up and running.

Installation Preparation

In this section, you will learn to prepare for Provisioning Gateway installation.

Installation Preparation includes downloading the required files and loading the files to the system. The steps are:

1. Download the Provisioning Gateway package (provgw-pkg-1.8.0.tgz) from Oracle Software Delivery Cloud (OSDC).
2. Untar the Provisioning Gateway package. Execute the following command to untar Provisioning Gateway Package File.
   `tar -xvf provgw-pkg-1.8.0.0.0.tgz`

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

   - **ProvGw Docker Images File:** `provgw-images-1.8.0.tar`
   - **Helm File:** `provgw-1.8.0.tgz`
   - **Readme txt File:** The `Readme.txt` contains cksum and md5sum of tarballs.
3. Verify the checksums of tarballs. Execute the following command:
   `cat Readme.txt`
4. Load the image tarballs to docker images. Execute the following command:
   `docker load --input provgw-images-1.8.0.tar`
5. Check if all the images are loaded. Execute the following command:
   `docker images | grep provgw`
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 Commands:

   ```
   docker tag provgw/provgw-service:1.8.0 <customer repo>/provgw-service:1.8.0
   docker tag provgw/auditor-service:1.8.0 <customer repo>/auditor-service:1.8.0
   docker tag provgw/ocingress_gateway:1.8.1 <customer repo>/ocingress_gateway:1.8.1
   docker tag provgw/configurationinit:1.4.0 <customer repo>/configurationinit:1.4.0
   docker tag provgw/configurationupdate:1.4.0 <customer repo>/configurationupdate:1.4.0
   docker tag provgw/ocegress_gateway:1.8.1 <customer repo>/ocegress_gateway:1.8.1
   ```
7. Push the docker images to docker registry. Execute the following command:
   `docker push <docker-repo>/<image-name>:<image-tag>`

   Sample Commands:

   ```
   docker push <customer repo>/provgw-service:1.8.0
   docker push <customer repo>/auditor-service:1.8.0
   docker push <customer repo>/ocingress_gateway:1.8.0
   docker push <customer repo>/configurationinit:1.4.0
   docker push <customer repo>/configurationupdate:1.4.0
   docker push <customer repo>/ocegress_gateway:1.8.1
   ```
8. Untar Helm Files. Execute the following command:

```
tar -xvzf provgw-1.8.0.tgz
```

9. Download the Provisioning Gateway 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 Provisioning Gateway Custom Template link to download the zip file.
   d. Unzip the template to get provgw-custom-configtemplates-1.8.0 file that contains the following:
      - **ProvGW_Dashboard.json:** This file is used by grafana.
      - **provgw-custom-values-1.8.0.yaml:** This file is used during installation.

Following are the Provisioning Gateway Images.

Pod	Image
<helm_release_name>-provgw-service	provgw/provgw_service
<helm_release_name>-prov-ingressgateway	provgw/ocingress_gateway provgw/configurationinit provgw/configurationupdate
<helm_release_name>-prov-egressgateway	provgw/ocegress_gateway provgw/configurationinit provgw/configurationupdate
<helm_release_name>-auditor-service	provgw/auditor-service

3
Using Auditor Service

In this section, you will learn about different auditor service report formats and generate auditor service reports.

Overview

The **Auditor Service** is deployed along with other services of Provisioning Gateway, at the time of Provisioning Gateway installation. You do not have to install it separately. The Auditor service image and helm charts are part of Provisioning Gateway installation package.

This service:

- Audits the subscriber data on SLF segments and creates a report if there is any discrepancy in the subscriber information.
- Reads the configuration information of key type and key values range to perform the audit.
- Connects to the configured SLF segments and sends REST requests to each one of the SLF via Egress Gateway.
- Compares the responses and determines if there are discrepancies.
- Creates a report in the PVC mounted path. This path contains subscriber key details that has different data on SLF segments.
- Continues with the next cycle of audit.

> **Note:**
>
> The Auditor Service uses PVC (Persistent Volume Claim) that you must create before installing Provisioning Gateway application.

Auditor Service Report

An Auditor Service generates a report when it finds any discrepency in the subscriber information. There are three types of report that it generates, which are as follows:

- **A subscriber exists in both the SLFs, but they have different data (key/ slfgroup)**

```
Responses do not match for msisdn: 1013001444 udr1-
ingressgateway.udr1:
{"profile-data":{"accountID":["10000000000000000013001444"],"imsi":
["1013001444"],
"msisdn":["1013001444"]},"slfGroupName":"LteGrp10"}. udr2-
ingressgateway.udr2:
```

{"profile-data":{"accountID":["10000000000000000000013001444"],"imsi":
["1013001444"],
"msisdn":["1013001444"]},"slfGroupName":"LteGrp09"}

Responses do not match for msisdn: 1013001446 udr1-
ingressgateway.udr1:
{"profile-data":{"accountID":["10000000000000000000013001446"],"imsi":
["1213001444"],
"msisdn":["1013001446"]},"slfGroupName":"LteGrp10"}. udr2-
ingressgateway.udr2:
{"profile-data":{"accountID":["10000000000000000000001300146"],"imsi":
["1013001446"],
"msisdn":["1013001446"]},"slfGroupName":"LteGrp09"}

- **Subscriber exists in one of the segment and not present in other**

Response not found for msisdn: 1013001458 udr1-ingressgateway.udr1:
Not Found. udr2-ingressgateway.udr2: {"profile-data":{"accountID":
["10000000000000000000013001458"],"imsi":["1013001458"],"msisdn":
["1013001458"]},
"slfGroupName":"LteGrp10"}
Response not found for msisdn: 1013001455 udr1-ingressgateway.udr1:
Not Found. udr2-ingressgateway.udr2: {"profile-data":{"accountID":
["10000000000000000000013001455"],"imsi":["1013001455"],"msisdn":
["1013001455"]},
"slfGroupName":"LteGrp10"}

Response not found for msisdn: 1013002788 udr1-ingressgateway.udr1:
{"profile-data":{"accountID":["10000000000000000000013000787"],"imsi":
["1013002788"],"msisdn":["1013002788"]},"slfGroupName":"LteGrp10"}.
udr2-ingressgateway.udr2: Not Found
Response not found for msisdn: 1013002787 udr1-ingressgateway.udr1:
{"profile-data":{"accountID":["10000000000000000000013000786"],"imsi":
["1013002787"],"msisdn":["1013002787"]},"slfGroupName":"LteGrp10"}.
udr2-ingressgateway.udr2: Not Found

- **Error Response from one of the SLFs**

Error response received for msisdn: 1013012788 udr1-
ingressgateway.udr1:
{"profile-data":{"accountID":["10000000000000000000013012788"],"imsi":
["1013012788"],
"msisdn":["1013012788"]},"slfGroupName":"LteGrp10"}. udr2-
ingressgateway.udr2: Error

Error response received for msisdn: 1013012789 udr1-
ingressgateway.udr1: Error.
udr2-ingressgateway.udr2: {"profile-data":{"accountID":
["10000000000000000000013012789"],
"imsi":["1013012789"],"msisdn":
["1013012789"]},"slfGroupName":"LteGrp10"}

Generating Auditor Service Report

Auditor Service reports generate continuously for files having below format:

`auditYYYYMMDDhhmm.txt`

Example: `audit202008190928.txt`

However, it is important to get a complete pod name of the Auditor Service. Execute the following command to get the complete pod name:

`kubectl get pods -n <provgw namespace>`

Example: `kubectl get pods -n provgw`

After identifying the Auditor Service pod name, you have three options to generate the report.

1. Login to the pod and execute the following command to view the report:
   `kubectl exec -it <complete pod name> -n <namespace > -- bash`

   This command opens a new shell in the respective pod. The necessary auditor reports are present in `/home/provuser/log/` path of the container. Execute the following command to navigate to the respective path:

   `cd /home/provuser/log/`

   Example: `kubectl exec -it provgw-auditor-service-8454f96f69-l5ddw -n provgw -- bash`

2. Execute the following command to copy the report log files to the local node:
   `kubectl cp <namespace>/<complete-pod-name>:/home/provuser/log/ /tmp/provLog/`

   This command creates a directory under `/tmp/provLog` location of the local node and copies all the audit files from the container to this directory.

   Example: `kubectl cp provgw/provgw-auditor-service-8454f96f69-l5ddw:/home/provuser/log/ /tmp/provLog/`

3. Monitor the auditor logs to identify a file and then, execute the following command to copy the file.
   `kubectl cp <namespace>/<complete-pod-name>:/home/provuser/log/<audit file name> /tmp/<audit file name>`

   Example: `kubectl cp provgw/provgw-auditor-service-8454f96f69-l5ddw:/home/provuser/log/audit202008190928.txt /tmp/provLog/`

4
Handling SLF Segments

In this section, you will learn about SLF Segments, detecting any failed SLF Segment in Provisioning Gateway setup, removing that failed SLF Segment and after fixing the issue, restoring that segment back to the setup.

Understanding the SLF Segment

The Provisioning Gateway receives provisioning requests from a provisioning client, forks the requests and sends it to all its configured SLF Segments. It accumulates all the responses it receives from each of the SLF Segment and sends a final response to the provisioning client.

> **Note:**
>
> The SLF Segment can have more than one SLFs. And, you can deploy more than one SLF Segment.

If all the SLFs in the SLF Segment are down and other segment responds success, the final response is failure. It causes the provisioning system to retry the same message without letting the user to know about the original problem. In this case, the user needs to remove that faulty segment from the Provisioning Gateway.

Detecting SLF Segment Failure

Provisioning Gateway raises Prometheus alerts that helps you to detect the SLF Segment failure or unavailability. You can monitor the following alerts using the Prometheus server:

- **ProvgwSegmentDownAbove1Percent (Warning):** Indicates more than 1% and less than 10% of the total number of messages lost due to a segment failure.
- **ProvgwSegmentDownAbove10Percent (Minor):** Indicates more than 10% and less than 25% of the total number of messages lost due to a segment failure.
- **ProvgwSegmentDownAbove25Percent (Major):** Indicates more than 25% and less than 50% of the total number of messages lost due to a segment failure.
- **ProvgwSegmentDownAbove50Percent (Critical):** Indicates 50% or more number of total messages lost due to segment failure.

[OR]

You can observe the Provisioning Gateway logs. These logs helps you to identify whether all the IP Addresses in the SLF Segment are down or not.

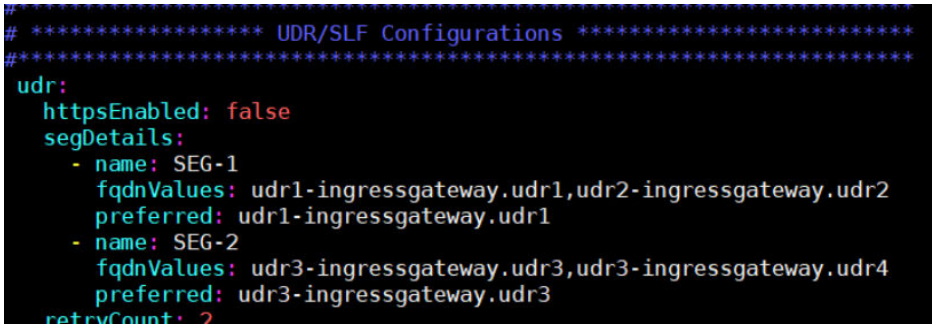
You can also receive or get the responses and verify which segment is not available or has gone down while looking into the problem statement.

Removing Failed SLF Segment

To remove the failed SLF Segment:

1. Execute the following command to copy the **provgw-custom-values.yaml** file.
   `cp provgw-custom-values.yaml provgw-custom-values.yaml.bkp`
2. Edit the **provgw-custom-values.yaml** file and remove the faulty segment details completely.
3. Stop the Auditor Service.
   Example:
   a. In the initial deployment, there are two segments, SEG-1. SEG-2.

Figure 4-1 Segments Deployed

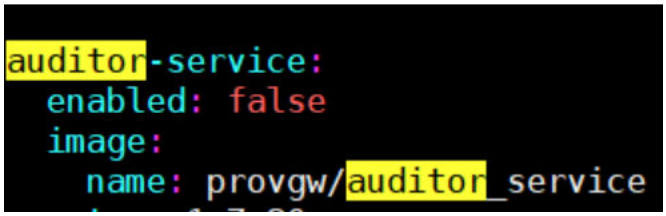


   b. The user identifies that SEG-1 has gone down and wants to remove it. To remove SEG-1, user should remove the entire segment configuration details from the .yaml file as shown below:

Figure 4-2 Removing Segment Details

   c. Change the **auditor-service.enabled** to false to stop the Auditor service.

Figure 4-3 Stopping Auditor Service



d. After making necessary configuration changes in the .yaml file, execute the following command to do Helm upgrade.
```
helm upgrade <release-name> <helm chart path> –- namespace <provgw namespace>
```

Restoring SLF Segment

After updating the SLF segment and assuring that it is ready to accept the messages, the user can add this segment back to the previous configuration. To restore the SLF Segment:

1. Copy the **provgw-custom-values.yaml.bkp** to **provgw-custom-values.yaml** if the updated SLF Segment is same as the previous one. If it is a new SLF Segment, then update the new segment details in the **provgw-custom-values.yaml** file.
2. After making necessary changes, execute the following command to upgrade helm:
```
helm upgrade <release-name> <helm chart path> – namespace <provgw namespace>
```
3. Verify if all the pods are up and running in the same namespace.

5
Customizing Provisioning Gateway

In this section, you will learn to customize Provisioning Gateway deployment. You can customize it by overriding the default values of various configurable parameters.

A Provisioning Gateway Customization file is given below:

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

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

# Configure customer created service accounts
  serviceAccountName:

# Configuration to enable UDR egress traffic through EGW
  egress:
    port: 8080
    enabled: "true"

  # 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)
  ingressGatewayHttpSignalingPort: 80
  ingressGatewayHttpsSignalingPort: 443

 #*********************************************************************
 # ***************** UDR/SLF Configurations ************************
 # *Please provde the SLF/UDR FQDNs persegment.
 # *If you have only one segment, make sure you have the
auditor_service.enable is set to false
 # *Allowed values: FQDNs, IP:Port, FQDN:Port
 #*********************************************************************
  udr:
    httpsEnabled: false
    segDetails:
      - name: SEG-1
        fqdnValues: udr1-ingressgateway.udr1,udr2-ingressgateway.udr2
        preferred: udr1-ingressgateway.udr1
      - name: SEG-2
        fqdnValues: udr3-ingressgateway.udr3,10.10.x.y:8081
        preferred: udr3-ingressgateway.udr3
    retryCount: 2
    connectTimeout: 10000
    connectionProbeTimer: 15000

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

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

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

  customExtension:
    allResources:
      labels: {}
      annotations: {}

    lbServices:
      labels: {}
      annotations: {}

    lbDeployments:
      labels: {}
      annotations: {}

    nonlbServices:
      labels: {}
      annotations: {}

    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
*************

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

# provgw-service microservice configurations
provgw-service:
  image:
    name: provgw/provgw_service
    tag: 1.8.0
```

```
    pullPolicy: Always

  service:
    type: ClusterIP
    port:
      https: 5002
      http: 5001
      management: 9000
    customExtension:
      labels: {}
      annotations: {}

  deployment:
    replicaCount: 2
    customExtension:
      labels: {}
      annotations: {}

  logging:
    level:
      root: "WARN"

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

  server:
    redirect:
      http: false
    http2enabled: true


#Application Specific configuration
  config:
    #retryErrorCodes : Transient error codes on which provgw will retry
the SLF requests
    retryErrorCodes: 500,503
    #retryCount: number of retries
    retryCount: 2
    #retryPeriod: time interval between each retry
    retryPeriod: 2

  minReplicas: 2
  maxReplicas: 4

# provgw-service microservice configurations
auditor-service:
  enabled: false
  image:
```

```
    name: provgw/auditor_service
    tag: 1.8.0
    pullPolicy: Always

  service:
    type: ClusterIP
    port:
      management: 9000
    customExtension:
      labels: {}
      annotations: {}

  deployment:
    replicaCount: 1
    customExtension:
      labels: {}
      annotations: {}

  logging:
    level:
      root: "INFO"

  resources:
    limits:
      cpu: 2
      memory: 2Gi
    requests:
      cpu: 2
      memory: 2Gi
    target:
      averageCpuUtil: 80

  server:
    redirect:
      http: false
    http2enabled: true

#Application Specific configuration
#This is mandatory for auditor application, Please provide the range of
subscribers to audit.
#The key must be either msisdn or imsi
  key:
    type: msisdn
    range: 1003000000-1003000200

  config:
    #Frequency between each audit
    auditFrequency: 15000
    #Throttle rate for SLF audit
    throttleRate: 100

  minReplicas: 1
  maxReplicas: 1
```

```
prov-ingressgateway:
  global:
    # Docker registry name

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

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

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

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

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

  deployment:
    customExtension:
      labels: {}
      annotations: {}

  service:
```

```
  ssl:
    tlsVersion: TLSv1.2
  customExtension:
    labels: {}
    annotations: {}

    privateKey:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      ecdsa:
        fileName: ecdsa_private_key_pkcs8.pem

    certificate:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: rovgw1
      rsa:
        fileName: apigatewayrsa.cer
      ecdsa:
        fileName: apigatewayecdsa.cer

    caBundle:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: caroot.cer

    keyStorePassword:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: key.txt

    trustStorePassword:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: trust.txt

    initialAlgorithm: RSA256

# Resource details
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
```

```
  target:
    averageCpuUtil: 80

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

# enable jaeger tracing
jaegerTracingEnabled: false

openTracing :
  jaeger:
    udpSender:
      # udpsender host
      host: "occne-tracer-jaeger-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

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

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

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

#OAUTH CONFIGURATION
oauthValidatorEnabled: false
nfType: SMF
nfInstanceId: 6faf1bbc-6e4a-4454-a507-a14ef8e1bc11
producerScope: nsmf-pdusession,nsmf-event-exposure
allowedClockSkewSeconds: 0
nrfPublicKeyKubeSecret: nrfpublickeysecret
nrfPublicKeyKubeNamespace: ingress
validationType: strict
producerPlmnMNC: 123
```

```
  producerPlmnMCC: 346

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

  maxRequestsQueuedPerDestination: 5000
  maxConnectionsPerIp: 10

  #The connectio TImeout must be greater than the requestTImeout
  connectionTimeout: 25000 #(ms)
  #The requestTImeout value must be greater than or equals to the the
product of config.retryPeriod and config.retryCount plus 5
  requestTimeout: 21000 #(ms)

  #Service Mesh (Istio) to take care of load-balancing
  serviceMeshCheck: false
  # configuring routes
  routesConfig:
  - id: traffic_mapping_rest_group_prov
    uri: http://{{ .Release.Name }}-provgw-service:5001
    path: /**
    order: 1


prov-egressgateway:
  #fullnameOverride : 'provgw-egress-gateway'
  nfType: ProvGw

  #global:
  #  dockerRegistry: udr-pv2-bastion-1:5000/ocudr

  deploymentEgressGateway:
    image: provgw/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

deployment:
  customExtension:
    labels: {}
    annotations: {}

openTracing :
  jaeger:
    udpSender:
      # udpsender host
      host: "jaeger-agent.cne-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
  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: 2
maxReplicas: 2
minAvailable: 5

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

service:
```

```
  type: ClusterIP
  customExtension:
    labels: {}
    annotations: {}
  ssl:
    tlsVersion: TLSv1.2
    initialAlgorithm: RSA256
    privateKey:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      ecdsa:
        fileName: ecdsa_private_key_pkcs8.pem

    certificate:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      rsa:
        fileName: apigatewayrsa.cer
      ecdsa:
        fileName: apigatewayecdsa.cer

    caBundle:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: caroot.cer

    keyStorePassword:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: key.txt

    trustStorePassword:
      k8SecretName: provgw-apigateway-secret
      k8NameSpace: provgw
      fileName: trust.txt
# ---- HTTPS Configuration - END ----

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

# Resource details
resources:
  limits:
    cpu: 3
    memory: 5Gi
    initServiceCpu: 1
    initServiceMemory: 1Gi
    updateServiceCpu: 1
    updateServiceMemory: 1Gi
  requests:
    cpu: 3
    memory: 5Gi
    initServiceCpu: 1
```

```
    initServiceMemory: 1Gi
    updateServiceCpu: 1
    updateServiceMemory: 1Gi
  target:
    averageCpuUtil: 80

#Set the root log level
log:
  level:
    root: WARN
    egress: INFO
    oauth: INFO
```

The configurable parameters of Provisioning Gateway are:

> **Note:**
>
> (*) - The fields in the following table tagged with '*" are mandatory.

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	
serviceAccountName	Service account name	null	Not Applicable	The serviceaccount, role and rolebindings required for deployment should be done prior to the installation. Use the created serviceaccountname here.
prefix.container	Container configurable prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for container names on different pods of ProvGw deployment. If Not configured, release name will be used as preifx.

parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
prefix.configmap	Configmap configurabe prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for configmap names. if Not configured, release name will be used as preifx.
prefix.hpa	HPA configurable prefix	null	Not Applicable	If this is configured with some value, the same will be used as prefix for HPA names. If Not configured, release name will be used as preifx.
egress.enabled	Enable egress gateway	true	true/false	This flag will enable egress gateway and all the requests to SLF will go via egress gateway from provisioning gateway
egress.port	Port of egress gateway	8080	Not Applicable	the https port of egress gateway which will send the requests to UDRs

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

*udr.segDetails	Segment details of UDRs	**Not Applicable	Not Applicable	To be used to send SLF requests to UDRs. This accepts yaml array of segments.
				name - Name of the segment
				fqdnValues - FQDNs/SLFs present in that segment
				preferred - preferred SLF among the fqdnValues
				e.g.
				`udr:`
				`segs:`
				`- name: SEG-1`
				`fqdnValues: ocudr1-ingressgateway.ocudr1,ocudr2-ingressgateway.ocudr2`
				`preferred: ocudr1-ingressgateway.ocudr1`
				`- name: SEG-2`
				`fqdnValues: ocudr3-ingressgateway.ocudr3,ocudr4-ingressgateway.ocudr4`
				`preferred: ocudr3-ingressgateway.ocudr3`

parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
udr.httpsEnabled	Enable https while sending requests UDR.	false	true/false	If UDR ingressgateway initssl and Incominghttps is enabled, and we need https only, then please enable this. Make sure the prov-egressgateway is deployed with initssl and enableOutGoing Https flags as true
udr.connectTimeout	The provgw timeout value for any request in case UDR/SLF doesn't respond in milliseconds	10000	Not Applicable	Time is in milliseconds
udr.connectionProbeTimer	Connection probe Timeout for periodic fetching of the active SLF in a segment in milliseconds	15000	Not Applicable	Time is in milliseconds
customExtension.allResources.labels	Custom Labels that needs to be added to all the OCNRF k8s resources	null	Not Applicable	This can be used to add custom label(s) to all k8s resources that will be created by OCNRF helm chart.
customExtension.allResources.annotations	Custom Annotations that needs to be added to all the OCNRF k8s resources	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 k8s resources that will be created by OCNRF helm chart.
customExtension.lbServices.labels	Custom Labels that needs to be added to OCNRF 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 OCNRF helm chart.

parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
customExtension.lbServices.annotations	Custom Annotations that needs to be added to OCNRF 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 OCNRF helm chart.
customExtension.lbDeployments.labels	Custom Labels that needs to be added to OCNRF 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 OCNRF helm chart which are associated to a Service which if of Load Balancer Type.
customExtension.lbDeployments.annotations	Custom Annotations that needs to be added to OCNRF Deployments that are associated to a Service which is 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 a custom annotation(s) to all Deployments that will be created by OCNRF helm chart which are associated to a Service which if of Load Balancer Type.
customExtension.nonlbServices.labels	Custom Labels that needs to be added to OCNRF 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 OCNRF helm chart.
customExtension.nonlbServices.annotations	Custom Annotations that needs to be added to OCNRF Services that are considered as not Load Balancer type	null	Not Applicable	This can be used to add a custom annotation(s) to all non-Load Balancer Type Services that will be created by OCNRF helm chart.

parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
customExtension.nonlbDeployments.labels	Custom Labels that needs to be added to OCNRF 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 label(s) to all Deployments that will be created by OCNRF helm chart which are associated to a Service which if not of Load Balancer Type.
customExtension.nonlbDeployments.annotations	Custom Annotations that needs to be added to OCNRF Deployments that are associated to a Service which is not of Load Balancer type	null	Not Applicable	This can be used to add custom annotation(s) to all Deployments that will be created by OCNRF helm chart which are associated to a Service which if not of Load Balancer Type.
k8sResource.container.prefix	Value that will be prefixed to all the container names of OCNRF.	null	Not Applicable	This value will be used to prefix to all the container names of OCNRF.
k8sResource.container.suffix	Value that will be suffixed to all the container names of OCNRF.	null	Not Applicable	This value will be used to prefix to all the container names of OCNRF.

Following table provides parameters for **provgw-service** micro service.

parameter	Description	Default value	Range of possible values(if applicable	Notes
image.pullPolicy	This setting will tell if image needs to be pulled or not	Always	Always IfNotPresent Never	
service.type	ProvGw service type	ClusterIP	ClusterIP NodePort LoadBalancer	The Kubernetes service type for exposing ProvGw deployment Note: Suggested to be set as ClusterIP (default value) always

parameter	Description	Default value	Range of possible values(if applicable	Notes
config.retryErrorCodes	Transient Error codes for retry	500,503	comma separated HTTP error codes	Upon receiving these transient error codes from UDR, provGw will retry with the same request to UDR.
image.name	Image name	provgw/provgw-service	Not Applicable	
image.tag	Tag of Image	1.8.0	Not Applicable	
service.port.http	HTTP port	5001	Not Applicable	The http port to be used in provGw service
service.port.https	HTTPS port	5002	Not Applicable	The https port to be used in provgw service
service.port.management	Management port	9000	Not Applicable	The Prometheus management port to be used for ProvGw service
deployment.replicaCount	Replicas of provgw pod	2	Not applicable	Number of provgw pods to be maintained by replica set created with deployment
config.retryCount	Retry count in case of transient error	2	Not applicable	Number of times retry should happen in case of transient error
config.retryPeriod	retry interval in seconds	2	Not applicable	The time gap between two retries. min value should be 1
resources.requests.cpu	Cpu Allotment for nudr-drservice pod	3	Not applicable	The cpu to be allocated for prov-gw pod during deployment
resources.requests.memory	Memory allotment for nudr-drservice pod	4Gi	Not applicable	The memory to be allocated for prov-gw pod during deployment
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	

parameter	Description	Default value	Range of possible values(if applicable	Notes
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
minReplicas	Minimum Replicas	2	Not Applicable	Minimum number of pods
maxReplicas	Maximum Replicas	4	Not Applicable	Maximum number of pods
service.customExtension.labels	Custom Labels that needs to be added to provgw specific Service.	null	Not applicable	This can be used to add custom label(s) to provgw Service.
service.customExtension.annotations	Custom Annotations that needs to be added to provgw specific Services.	null	Not applicable	This can be used to add custom annotation(s) to provgw Service.
deployment.customExtension.labels	Custom Labels that needs to be added to provgw specific Deployment.	null	Not applicable	This can be used to add custom label(s) to provgw Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to provgw specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to provgw Deployment.
server.redirect.http	Enable redirecting HTTP mesagases	false	true/false	
server.http2enabled	Enabled HTTP2 support flag	true	true/false	
logging.level.root	Log Level	WARN	WARN INFO DEBUG ERROR	Log level of the Provisioning gateway pod

Following table provides parameters for **auditor-service** micro service.

Parameter	Description	Default Value	Range of possible values (if applicable	Notes
enable	Enable/disable auditor service	false	true/false	This flag enables or disables auditor service
image.name	Image name	provgw/auditor-service	Not Applicable	
image.tag	Tag of Image	1.8.0	Not Applicable	

Parameter	Description	Default Value	Range of possible values (if applicable	Notes
image.pullPolicy	This setting will tell if the image needs to be pulled or not	Always	Always IfNotPresent Never	
service.type	ProvGw service type	ClusterIP	ClusterIP NodePort LoadBalancer	The Kubernetes service type for exposing ProvGw deployment Note: Suggested to be set as ClusterIp (default value) always
deployment.replicaCount	Replicas of auditor pod	1	Not applicable	Number of auditor pods to be maintained by replica set created with deployment
logging.level.root	Log Level	INFO	WARN INFO DEBUG ERROR	Log level of the auditor pod
key.type	type of key to be used for auditing	msisdn	msisdnimsi	
key.range	Range of keys to be audited	Not applicable	Not applicable	
config.auditFrequency	the frequency at which audit will start reporting again after completion of the previous instance	15000	Not applicable	time in milliseconds
config.throttleRate	Throttling rate for the auditor microservice to send messages to udr	100	1 - 2000	The total number of messages throttled per second to each slf
server.redirect.http	Enable redirecting HTTP mesagases	false	true/false	
server.http2enabled	Enabled HTTP2 support flag	true	true/false	
resources.requests.cpu	Cpu Allotment for nudr-drservice pod	2	Not applicable	The cpu to be allocated for auditor pod during deployment

Parameter	Description	Default Value	Range of possible values (if applicable	Notes
resources.requests.memory	Memory allotment for nudr-drservice pod	2Gi	Not applicable	The memory to be allocated for auditor pod during deployment
resources.limits.cpu	Cpu allotment limitation	2	Not applicable	
resources.limits.memory	Memory allotment limitation	2Gi	Not applicable	
resources.target.averageCpuUtil	CPU utilization limit for autoscaling	80	Not Applicable	CPU utilization limit for creating HPA
minReplicas	Minimum Replicas	1	Not Applicable	Minimum number of pods
maxReplicas	Maximum Replicas	1	Not Applicable	Maximum number of pods
service.customExtension.labels	Custom Labels that needs to be added to auditor specific Service.	null	Not applicable	This can be used to add custom label(s) to auditor Service.
service.customExtension.annotations	Custom Annotations that needs to be added to auditor specific Services.	null	Not applicable	This can be used to add custom annotation(s) to auditor Service.
deployment.customExtension.labels	Custom Labels that needs to be added to auditor specific Deployment.	null	Not applicable	This can be used to add custom label(s) to auditor Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to auditor specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to auditor Deployment.

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

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
global.type	provgw-prov-ingressgateway service type	LoadBalancer	Possbile Values- ClusterIP NodePort LoadBalancer	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
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.publicHttpSignalingPort	Port used on which ingressgateway listens for incoming http requests.	80	Valid Port	
global.publicHttpsSignallingPort	Port used on which ingressgateway listens for incoming https requests.	443	Valid Port	
image.name	Docker image name	provgw/ocingress_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	provgw/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	provgw/configurationupdate	Not applicable	
updateContainersImage.tag	Image version tag	1.4.0	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
updateContainersImage.pullPolicy	This setting will tell if image need to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
service.ssl.privateKey.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.privateKey.k8NameSpace	namespace in which secret is created	provgw	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	provgw-apigateway-secret	Not applicable	
service.ssl.certificate.k8NameSpace	namespace in which secret is created	provgw	Not applicable	
service.ssl.certificate.rsa.fileName	rsa certificate stored in the secret	apigatewayrsa.cer	Not applicable	
service.ssl.certificate.ecdsa.fileName	ecdsa certificate stored in the secret	apigatewayecdsa.cer	Not applicable	
service.ssl.caBundle.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.caBundle.k8NameSpace	namespace in which secret is created	provgw	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	provgw-apigateway-secret	Not applicable	
service.ssl.keyStorePassword.k8NameSpace	namespace in which secret is created	provgw	Not applicable	
service.ssl.keyStorePassword.fileName	keyStore password stored in the secret	key.txt	Not applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
service.ssl.trustStorePassword.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.trustStorePassword.k8NameSpace	namespace in which secret is created	provgw	Not applicable	
service.ssl.trustStorePassword.fileName	trustStore password stored in the secret	trust.txt	Not applicable	
resources.limits.cpu	Cpu allotment limitation	3	Not applicable	
resources.limits.memory	Memory allotment limitation	4Gi	Not applicable	
resources.limits.initServiceCpu	Maximum amount of CPU that **K8s** will allow the ingress-gateway init container to use.	1	Not applicable	
resources.limits.initServiceMemory	Memory Limit for ingress-gateway init container	1Gi	Not applicable	
resources.limits.updateServiceCpu	Maximum amount of CPU that K8s 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 provgw-prov-ingressgateway pod	3	Not Applicable	
resources.requests.memory	Memory allotment for provgw-prov-ingressgateway pod	4Gi	Not Applicable	
resources.requests.initServiceCpu	The amount of CPU that the system will guarantee for the ingress-gateway init container, and K8s 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.initServiceMemory	The amount of memory that the system will guarantee for the ingress-gateway init container, and K8s 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 ingress-gateway update container, and K8s 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 K8s 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 provgw-prov-ingressgateway pod	WARN	valid level	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
log.level.ingress	Logs to be shown on provgw-prov-ingressgateway pod for ingress related flows	INFO	valid level	
log.level.oauth	Logs to be shown on provgw-prov-ingressgateway pod for oauth related flows	INFO	valid level	
initssl	To Initialize SSL related infrastructure in init/update container	true	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.
oauthValidatorEnabled	OAUTH Configuration	false	Not Applicable	
enableIncomingHttp	Enabling for accepting http requests	true	Not Applicable	
enableIncomingHttps	Enabling for accepting https requests	true	true or false	
enableOutgoingHttps	Enabling for sending https requests	false	true or false	
maxRequestsQueuedPerDestination	Queue Size at the provgw-prov-ingressgateway pod	5000	Not Applicable	
maxConnectionsPerIp	Connections from ingressgateway to other microServices	10	Not Applicable	

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
routesConfig	Routes configured to connect to ProvGw	- id: traffic_mapping_rest_group_prov uri: http://{{ .Release.Name }}-prov-gw:5001 path: /**	Not Applicable	
service.customExtension.labels	Custom Labels that needs to be added to ingress-gateway specific Service.	null	Not applicable	This can be used to add custom label(s) to ingress-gateway Service.
service.customExtension.annotations	Custom Annotations that needs to be added to ingress-gateway specific Services.	null	Not applicable	This can be used to add custom annotation(s) to ingress-gateway Service.
deployment.customExtension.labels	Custom Labels that needs to be added to ingress-gateway specific Deployment.	nul	Not applicable	This can be used to add custom label(s) to ingress-gateway Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to ingress-gateway 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 ingress-gateway Deployment.
connectionTimeout	Timeout for each connection request	25000	Not applicable	This is used for configuring the timeout value for each client connection. This value must be greater than the requestTimeout
requestTimeout	TImeout for each request	21000	Not applicable	This config is used for configuring the request time out value. This must be greater than the product of config.retryCount and config.retryPeriod from provgw micro service

Parameter	Description	Default value	Range or Possible Values (If applicable)	Notes
serviceMeshCheck	Load balancing will be handled by Ingress gateway, if true it would be handled by serviceMesh	true	true/false	

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

Parameter	Description	Default Value	Range or Possible Values(if applicable)	Notes
type	provgw-prov-egressgateway service type	LoadBalancer	Possbile Values- ClusterIP NodePort LoadBalance	
image.name	Docker Image name	provgw/ocegress_gateway	Not applicable	
image.tag	Image version tag	1.8.1	Not applicable	
image.pullPolicy	This setting will tell if the image needs to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
initContainersImage.name	Docker Image name	provgw/configurationinit	Not applicable	
initContainersImage.tag	Image version tag	1.4.0	Not applicable	
initContainersImage.pullPolicy	This setting will tell if the image needs to be pulled or not	Always	Possible Values - Always IfNotPresent Never	
updateContainersImage.name	Docker Image name	provgw/configurationupdate	Not applicable	
updateContainersImage.tag	Image version tag	1.4.0	Not applicable	
updateContainersImage.pullPolicy	This setting will tell if the image needs to be pulled or not	Always	Possible Values - Always IfNotPresent Never	

Parameter	Description	Default Value	Range or Possible Values(if applicable)	Notes
service.ssl.privateKey.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.privateKey.k8NameSpace	namespace in which secret is created	provgw	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 secre	ecdsa_private_key_pkcs8.pem	Not applicable	
service.ssl.certificate.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.certificate.k8NameSpace	namespace in which secret is created	provgw	Not applicable	
service.ssl.certificate.rsa.fileName	rsa certificate stored in the secret	apigatewayrsa.cer	Not applicable	
service.ssl.certificate.ecdsa.fileName	ecdsa certificate stored in the secret	apigatewayecdsa.cer	Not applicable	
service.ssl.caBundle.k8SecretName	name of the secret which stores keys and certificates	provgw-apigateway-secret	Not applicable	
service.ssl.caBundle.k8NameSpace	namespace in which secret is created	provgw	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	provgw-apigateway-secret	Not applicable	
service.ssl.keyStorePassword.k8NameSpace	namespace in which secret is created	provgw	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	provgw-apigateway-secret	Not applicable	

Parameter	Description	Default Value	Range or Possible Values(if applicable)	Notes
service.ssl.trustStorePassword.k8NameSpace	namespace in which secret is created	provgw	Not applicable	
service.ssl.trustStorePassword.fileName	trustStore password stored in the secret	trust.txt	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-egressgateway pod	WARN	Not applicable	
log.level.egress	Logs to be shown on ocudr-egressgateway pod for egress related flows	INFO	Not applicable	
log.level.oauth	Logs to be shown on ocudr-egressgateway pod for oauth related flows	INFO	Not applicable	
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 K8s 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 K8s 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 provgw-prov-egressgateway pod	3	Not Applicable	
resources.requests.memory	Memory allotment for provgw-prov-egressgateway pod	4Gi	Not Applicable	
resources.requests.initServiceCpu	The amount of CPU that the system will guarantee for the egress-gateway init container, and K8s 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 K8s** 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 K8s** 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 K8s** 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	
openTracing.jaeger.probabilisticSampler	Probabilistic 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/false	
oauthClientEnabled	Enable if oauth is required	false	true/false	Enable based on Oauth configuration
nrfAuthority	Nrf Authoriy configuration	10.75.224.7:8085	Not Applicable	
nfInstanceId	Nrf Instance Id		Not Applicable	
consumerPlmnMNC	plmnmnc	345	Not Applicable	
consumerPlmnMCC	plmnmcc	567	Not Applicable	
service.customExtension.labels	Custom Labels that needs to be added to egress-gateway specific Service.	null	Not applicable	This can be used to add custom label(s) to ingress-gateway Service.
service.customExtension.annotations	Custom Annotations that needs to be added to egress-gateway specific Services.	null	Not applicable	This can be used to add custom annotation(s) to egress-gateway Service.

Parameter	Description	Default Value	Range or Possible Values(if applicable)	Notes
deployment.customExtension.labels	Custom Labels that needs to be added to egress-gateway specific Deployment.	null	Not applicable	This can be used to add custom label(s) to egress-gateway Deployment.
deployment.customExtension.annotations	Custom Annotations that needs to be added to egress-gateway specific Deployment.	null	Not applicable	This can be used to add custom annotation(s) to egress-gateway Deployment.

Provisioning Gateway KPIs and Metrics

In this section, you will learn about KPIs and Metrics of Provisioning Gateway.

Ingress API Gateway Metrics

Metric	Metric Details	Service Operation	Response Code	Notes
Total Ingress Requests	oc_ingressgateway_http_requests_total	All	Not Applicable	oc_ingressgateway_http_requests_total
Total Ingress Responses	oc_ingressgateway_http_responses_total	All	Not Applicable	oc_ingressgateway_http_responses_total
Request with 2xx Responses	Total no of success response with status code as 2xx	Get,Put & Delete Operations	2xx	oc_ingressgateway_http_responses_total{Status=~"2.*"} (If you want method specific 2xx response count, then pass the method value as one of the parameter. Sample to get 2xx GET response count is: oc_ingressgateway_http_responses_total{Status=~"2.*",Method="GET"})

Metric	Metric Details	Service Operation	Response Code	Notes
Request with 4xx Responses	Total no of success response with status code as 4xx	Get,Put & Delete Operations	4xx	oc_ingressgateway_http_responses_total{Status=~"4.*"} (If you want method specific 4xx response count, then pass the method value as one of the parameter. Sample to get 4xx GET response count is: oc_ingressgateway_http_responses_total{Status=~"4.*",Method="GET"})
Request with 5xx Responses	Total no of success response with status code as 5xx	Get,Put & Delete Operations	5xx	oc_ingressgateway_http_responses_total{Status=~"5.*"} (If you want method specific 4xx response count, then pass the method value as one of the parameter. Sample to get 4xx GET response count is: oc_ingressgateway_http_responses_total{Status=~"5.*",Method="GET"})

Egress API Gateway Metrics

Metric	Metric details	service operation	response code	notes
Total Egress Requests	oc_egressgateway_http_requests_total			
Total Egress Responses	oc_egressgateway_http_responses_total			

Metric	Metric details	service operation	response code	notes
Number of Requests sent by Provgw with PUT towards other NF's	Total number of PUT requests sent by provgw	PutRequests	201	oc_egressgateway_http_requests_total{Method="PUT"}
Number of Requests sent by provgw with GET towards other NF's	Total number of GET requests sent by provgw	GetRequests	200	oc_egressgateway_http_requests_total{Method="GET"}
Number of Requests sent by provgw with DELETE towards other NF's	Total number of DELETE requests sent by provgw	DeleteRequests	204	oc_egressgateway_http_requests_total{Method="DELETE"}
PutRequests with 2xx	Total number of success PUT response with status as 2xx	PutRequests	2xx	oc_egressgateway_http_responses_total{Method="PUT",Status=~"2.*"}
GetRequests with 2xx	Total number of success GET response with status as 2xx	GetRequests	2xx	oc_egressgateway_http_responses_total{Method="GET",Status=~"2.*"}
DeleteRequests with 2xx	Total number of success DELETE response with status as 2xx	DeleteRequests	2xx	oc_egressgateway_http_responses_total{Method="DELETE",Status=~"2.*"}
PutRequests with 4xx	Total number of failure PUT response with status as 4xx	PutRequests	4xx	oc_egressgateway_http_responses_total{Method="PUT",Status=~"4.*"}
GetRequests with 4xx	Total number of failure GET response with status as 4xx	GetRequests	4xx	oc_egressgateway_http_responses_total{Method="GET",Status=~"4.*"}
DeleteRequests with 4xx	Total number of failure DELETE response with status as 4xx	DeleteRequests	4xx	oc_egressgateway_http_responses_total{Method="DELETE",Status=~"4.*"}

Metric	Metric details	service operation	response code	notes
No of HTTP request sent by provgw	Total number of HTTP requests sent by provgw. If you want a method specific, then add a method as one more tag as a comma-separated.	All	Not Applicable	oc_egressgateway_http_requests_total{Scheme="HTTP"}

Provisioning Gateway Metrics

Category	Sub-Category	Description	Metrics Name	Notes
Rest Controller ProvGw	Ingress	provGwRestRequestTotal	provGw_rest_request{Method=""}	Gives total number of request received by the ProvGw when you pass the Method value as blank. If you want to get the total number of GET or PUT or DELETE request recieved then pass the same as the value of the Method. Sample to check total no. of GET request: provGw_rest_request{Method="GET"}

Category	Sub-Category	Description	Metrics Name	Notes
		provGwRestResponseTotal	provGw_rest_response{Method=""}	Gives total number of response send from the ProvGw when you pass the Method value as blank. If you want to get the total number of GET or PUT or DELETE response sent then you have to pass the same as the value of the Method. Sample to check total no. of GET request: provGw_rest_response{Method="GET"}
		provGwRestSuccessResponse	provGw_rest_successResponse{Method=""}	Gives total number of successful response send from ProvGw when you pass the Method value as blank. If you want the total number of GET or PUT or DELETE success response then pass the same as the value of the Method. Sample to check total number of GET success response is: provGw_rest_successResponse{Method="GET"}

Category	Sub-Category	Description	Metrics Name	Notes
		provGwRestFailureResponse	provGw_rest_failureResponse{Method=""}	Gives total number of failure response send from ProvGw when you pass the Method value as blank. If you want the total number of GET or PUT or DELETE failure response then pass the same as the value of the Method. Sample to check the total number of GET failure response: provGw_rest_failureResponse{Method="GET"}

Category	Sub-Category	Description	Metrics Name	Notes
	Egress	udrRestRequestTotal	udr_rest_request{Method="", udrSegment="SEG NAME"}	Gives the total number of request send to the UDR (segment-1) when you pass the Method value as blank and the udrSegment value as- name of UDR segment-1. Similarly if you want to check the same for UDR segment-2, pass the udrSegment value as- name of UDR segment-2. If you want to check the total number of GET or PUT or DELETE request send to udr segment-1 or udr segment-2, then pass the same as the value of the Method and udrSegment as SEG-1 NAME/ SEG-2 NAME. Sample to check total number of GET request sent to udr segment 1 is: udr_rest_request{Method="GET", udrSegment="SEG-1 NAME"} Sample to check total number of GET request sent to udr segment 2 is: udr_rest_request{Method="GET", udrSegment="SEG-2 NAME"}

Category	Sub-Category	Description	Metrics Name	Notes
		udrRestResponseTotal	udr_rest_response{Method="", udrSegment = "SEG NAME"}	Gives the total number of response received from UDR(segment-1) when you pass the udrSgement value as- name of UDR segment-1. Similarly if you want to check the same for UDR segment-2, then pass the udrSegment value as- name of UDR segment-2. If you want to check the total number of GET or PUT or DELETE response received from udr segment-1 or udr segment-2 , you have to pass the same, as the value of the Method and udrSegment as SEG-1 NAME/ SEG-2 NAME. Sample to check total number of GET response received from udr segment 1: udr_rest_response{Method="GET", udrSegment="SEG-1 NAME"} Sample to check total number of GET response received from udr segment 2: udr_rest_response{Method="GET", udrSegment="SEG-2 NAME"}

Category	Sub-Category	Description	Metrics Name	Notes
		udrRestSuccessResponse	udr_rest_successResponse{Method="", udrSegment="SEG NAME"}	Gives the total number of success response received from the UDR (segment-1) when you pass the Method value as blank and the udrSegment value as- name of the udr segment-1. Similarly if you want to check the same for UDR segment-2, pass the udrSegment value as- name of udr segment-2. If you want to check the total number of GET or PUT or DELETE success response received from udr segment-1 or udr segment-2, then pass the same as the value of the Method and udrSegment as SEG-1 NAME/ SEG-2 NAME. Sample to check total number of GET success response received from udr segment 1 is: udr_rest_successResponse{Method="GET", udrSegment="SEG-1 NAME"} Sample to check the total number of GET success response received from udr segment 2 is: udr_rest_successResponse{Method="GET", udrSegment="SEG-2 NAME"}

Category	Sub-Category	Description	Metrics Name	Notes
		udrRestFailureResponse	udr_rest_failureResponse{Method="", udrSegment="SEG-1 NAME"}	Gives the total number of failure response received from the UDR (segment-1), when you pass the Method value as blank and the udrSegment value as name of the udr segment-1. Similarly if you want to check the same for UDR segment-2, you have to pass the udrSegment value as name of udr segment-2. If you want to check the total number of GET or PUT or DELETE failure response received from udr segment-1 or udr segment-2 , then pass the same as the value of the Method and udrSegment as SEG-1 NAME/ SEG-2 NAME. Sample to check total number of GET failure response received from udr segment 1 is: udr_rest_failureResponse{Method="GET", udrSegment="SEG-1 NAME"} Sample to check total number of GET success response received from udr segment 2 is: udr_rest_failureResponse{Method="GET",

Category	Sub-Category	Description	Metrics Name	Notes
				udrSegment="SEG-2 NAME"}
	Egress	Transient error metrics	udr_rest_transient_error	Gives the total number of transient errors received. Currently, only 5xx errors are considered as transient errors. This does not need any parameter.
	Egress	Service not available/ segment down	udr_rest_service_unavailable	Gives the total number of messages that have failed even with retries to a segment. This is a potential indication of a segment becoming unavailable. This does not need any parameter.

Auditor Metrics

Category	Sub-Category	Description	Metrics Name	Notes
	Egress	Successful audit with same content in both segments	provgw_audit_success	Total number of successful auditing where the responses received from both the SLFs matches.
		Successful audit with different content in both segments	provgw_audit_responsemismatch	Total number of successful auditing where the responses received from both the SLFs does not match.
		total failure count	provgw_auditor_failure	Total number of audit failures where auditor failed to validate/ audit the responses.

Category	Sub-Category	Description	Metrics Name	Notes
		total number of subscriber audited	provgw_audit_total	Total number of subscriber audits carried out by auditor microservice.
	Egress	Transient error metrics	provgw_audit_transient_error	Gives the total number of transient errors received. currently. Only 5xx errors are considered as transient errors. This does not need any parameter.
	Egress	Service not available/ segment down	provgw_audit_service_unavailable	Gives the total number of messages that have failed even with retries to a segment. This is a potential indication of a segment becoming unavailable. This does not need any parameter.

KPIs - provgw-ingress-gateway

KPI Details	Service Operation	KPI	Response Code	Notes
No of Requests/sec	All	ProvGw Ingress Request Rate	Not Applicable	sum(irate(oc_ingressgateway_http_requests_total[5m]))
No of Responses/sec	All	ProvGw Ingress Response Rate	Not Applicable	sum(irate(oc_ingressgateway_http_responses_total[5m]))

KPI Details	Service Operation	KPI	Response Code	Notes
No of 2xx responses per second	GET,PUT & DELETE Request	rate of Requests with 2xx response code	200/201/204	sum(irate(oc_ingressgateway_http_responses_total{Status=~"2.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])) (If you want method specific 2xx response rate, then pass it as one of the parameters, Sample to check the rate of GET requests with 2xx response is: sum(irate(oc_ingressgateway_http_responses_total{Method="GET", Status=~"2.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])))

KPI Details	Service Operation	KPI	Response Code	Notes
No of 4xx responses per second	GET,PUT & DELETE Request	rate of Requests with 4xx response code	400/404	sum(irate(oc_ingressgateway_http_responses_total{Status=~"4.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])) (If you want method specific 4xx response rate, then pass it as one of the parameters, Sample to check the rate of GET requests with 4xx response is: sum(irate(oc_ingressgateway_http_responses_total{Method="GET", Status=~"4.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])))

KPI Details	Service Operation	KPI	Response Code	Notes
No of 5xx responses per second	GET,PUT & DELETE Request	rate of Requests with 5xx response code	500/503	sum(irate(oc_ingressgateway_http_responses_total{Status=~"5.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])) (If you want method specific 5xx response rate, then pass it as one of the parameters, Sample to check the rate of GET requests with 5xx response is: sum(irate(oc_ingressgateway_http_responses_total{Method="GET", Status=~"5.*"}[5m]))/ sum(irate(oc_ingressgateway_http_responses_total[5m])))

6
Configuring Alerts

To configure Provisioning Gateway alerts on the Prometheus server:

> **Note:**
>
> In the below procedure, _**NAME**_ is the Helm Chart Release Name and _**Namespace**_ is the Prometheus NameSpace.

1. Execute the following command to take the backup of current config map of Prometheus.
   ```
   kubectl get configmaps occne-prometheus-server -o yaml -n occne-infra
   > /tmp/tempConfig.yaml
   ```
2. Check and add provisioning gateway alert file name inside Prometheus config map as shown below:

   ```
   sed -i '/etc\/config\/alertsprovgw/d' /tmp/tempConfig.yaml
   sed -i '/rule_files:/a\    \- /etc/config/alertsprovgw'  /tmp/
   tempConfig.yaml
   ```

3. Execute the following command to update the config map with updated file name of provgw alert file.
   ```
   kubectl replace configmap occne-prometheus-server -f /tmp/
   tempConfig.yaml
   ```
4. Execute the following command to add provgw alert rules in config map under file name of provgw alert file.
   ```
   kubectl patch configmap occne-prometheus-server -n occne-infra --type
   merge --patch "$(cat ~/ProvgwAlertrules.yaml)"
   ```

   > **Note:**
   >
   > Prometheus server takes updated config map, which reloads automatically after sometime (~20 sec).

Provisioning Gateway Alert Config Details

This section shares the alert config details of the **ProvgwAlertrules.yaml** file.

> **Note:**
>
> The default nameSpace of Provisioning Gateway is provgw. Update it according to the deployment.

```
apiVersion: v1
data:
  alertsudr: |
    groups:
    - name: ProvgwAlerts
      rules:
      - alert: ProvgwTrafficRateAboveMinorThreshold
        annotations:
          description: 'Ingress traffic Rate is above minor threshold
i.e. 800 requests
          per second (current value is: {{ $value }})'
          summary: 'Traffic Rate is above 80 Percent of Max requests
per second(1000)'
        expr:
sum(rate(oc_ingressgateway_http_requests_total{app_kubernetes_io_name=
        "ingressgateway",kubernetes_namespace="provgw"}[20m])) >= 800 <
900
        labels:
          severity: Minor
      - alert: ProvgwTrafficRateAboveMajorThreshold
        annotations:
          description: 'Ingress traffic Rate is above major threshold
i.e. 900 requests
          per second (current value is: {{ $value }})'
          summary: 'Traffic Rate is above 90 Percent of Max requests
per second(1000)'
        expr:
sum(rate(oc_ingressgateway_http_requests_total{app_kubernetes_io_name=
        "ingressgateway",kubernetes_namespace="provgw"}[20m])) >= 900 <
950
        labels:
          severity: Major
      - alert: ProvgwTrafficRateAboveCriticalThreshold
        annotations:
          description: 'Ingress traffic Rate is above critical
threshold i.e. 950 requests
          per second (current value is: {{ $value }})'
          summary: 'Traffic Rate is above 95 Percent of Max requests
per second(1000)'
        expr:
sum(rate(oc_ingressgateway_http_requests_total{app_kubernetes_io_name=
        "ingressgateway",kubernetes_namespace="provgw"}[20m])) >= 950
        labels:
          severity: Critical
      - alert: ProvgwTransactionErrorRateAbove0.1Percent
        annotations:
          description: 'Transaction Error rate is above 0.1 Percent of
Total Transactions
```

```
         (current value is {{ $value }})'
          summary: 'Transaction Error Rate detected above 0.1 Percent
of Total
          Transactions'
        expr: (sum(rate(oc_ingressgateway_http_responses_total{Status!
~"2.*",

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m])
        or (up * 0 ) ) )/
sum(rate(oc_ingressgateway_http_responses_total{

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m]))
        * 100 >= 0.1 < 1
        labels:
          severity: Warning
      - alert: ProvgwTransactionErrorRateAbove1Percent
        annotations:
          description: 'Transaction Error rate is above 1 Percent of
Total Transactions
        (current value is {{ $value }})'
          summary: 'Transaction Error Rate detected above 1 Percent of
Total Transactions'
        expr: (sum(rate(oc_ingressgateway_http_responses_total{Status!
~"2.*",

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m])
        or (up * 0 ) ) )/sum(rate(oc_ingressgateway_http_responses_total

{app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m]))
        * 100 >= 1 < 10
        labels:
          severity: Warning
      - alert: ProvgwTransactionErrorRateAbove10Percent
        annotations:
          description: 'Transaction Error rate is above 10 Percent of
Total Transactions
          (current value is {{ $value }})'
          summary: 'Transaction Error Rate detected above 10 Percent of
Total
          Transactions'
        expr: (sum(rate(oc_ingressgateway_http_responses_total{Status!
~"2.*",

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
        [20m]) or (up * 0 ) ) )/
sum(rate(oc_ingressgateway_http_responses_total

{app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m]))
         * 100 >= 10 < 25
        labels:
```

```
          severity: Minor
      - alert: ProvgwTransactionErrorRateAbove25Percent
        annotations:
          description: 'Transaction Error Rate detected above 25
Percent of Total
        Transactions (current value is {{ $value }})'
          summary: 'Transaction Error Rate detected above 25 Percent of
Total
        Transactions'
        expr: (sum(rate(oc_ingressgateway_http_responses_total{Status!
~"2.*",

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m])
        or (up * 0 ) ) )/sum(rate(oc_ingressgateway_http_responses_total

{app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m])) *
         100 >= 25 < 50
        labels:
          severity: Major
      - alert: ProvgwTransactionErrorRateAbove50Percent
        annotations:
          description: 'Transaction Error Rate detected above 50
Percent of Total
        Transactions (current value is {{ $value }})'
          summary: 'Transaction Error Rate detected above 50 Percent of
Total
        Transactions'
        expr: (sum(rate(oc_ingressgateway_http_responses_total{Status!
~"2.*",

app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m]) or
        (up * 0 ) ) )/sum(rate(oc_ingressgateway_http_responses_total

{app_kubernetes_io_name="ingressgateway",kubernetes_namespace="provgw"}
[20m]))
        * 100 >= 50
        labels:
          severity: Critical
      - alert: ProvgwTransientErrorAbove1Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 1% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 1% of
        ingress traffic'
        expr:
(sum(rate(udr_rest_transient_error{kubernetes_namespace="provgw"}[10m]))
        /
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace="pro
vgw"}
        [10m])))*100 >= 1 < 10
```

```
        labels:
          severity: Warning
      - alert: ProvgwTransientErrorAbove10Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 10% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 10% of
        ingress traffic'
        expr:
(sum(rate(udr_rest_transient_error{kubernetes_namespace="provgw"}
[10m]))/

sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace="pro
vgw"}
        [10m])))*100 >= 10 < 25
        labels:
          severity: Minor
      - alert: ProvgwTransientErrorAbove25Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 25% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 25% of
          ingress traffic'
        expr:
(sum(rate(udr_rest_transient_error{kubernetes_namespace="provgw"}[10m]))
        /
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace="pro
vgw"}
        [10m])))*100 >= 25 < 50
        labels:
          severity: Major
      - alert: ProvgwTransientErrorAbove50Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(udr_rest_transient_error{kubernetes_namespace="provgw"}[10m]))
        /
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace="pro
vgw"}
        [10m])))*100 >= 50
        labels:
          severity: Critical
      - alert: ProvgwSegmentDownAbove1Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
```

```
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(udr_rest_service_unavailable{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace=
        "provgw"}[10m])))*100 >= 1 < 10
        labels:
          severity: Warning
      - alert: ProvgwSegmentDownAbove10Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(udr_rest_service_unavailable{kubernetes_namespace="provgw"}
[10m]))
        /
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace="pro
vgw"}
        [10m])))*100 >= 10 < 25
        labels:
          severity: Minor
      - alert: ProvgwSegmentDownAbove25Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(udr_rest_service_unavailable{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace=
        "provgw"}[10m])))*100 >= 25 < 50
        labels:
          severity: Major
      - alert: ProvgwSegmentDownAbove50Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(udr_rest_service_unavailable{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(oc_ingressgateway_http_requests_total{kubernetes_namespace=
```

```
        "provgw"}[10m])))*100 >= 50
        labels:
          severity: Critical
      - alert: ProvgwAuditMismatchAbove1Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(provgw_audit_responsemismatch{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
        >= 1 < 10
        labels:
          severity: Warning
      - alert: ProvgwAuditMismatchAbove10Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(provgw_audit_responsemismatch{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
>=
         10 < 25
        labels:
          severity: Minor
      - alert: ProvgwAuditMismatchAbove25Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(provgw_audit_responsemismatch{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
        >= 25 < 50
        labels:
          severity: Major
      - alert: ProvgwAuditMismatchAbove50Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
```

```
about 50% of
        ingress traffic'
        expr:
(sum(rate(provgw_audit_responsemismatch{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
>= 50
        labels:
          severity: Critical
      - alert: ProvgwAuditTransientErrorAbove1Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(provgw_audit_transient_error{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
        >= 1 < 10
        labels:
          severity: Warning
      - alert: ProvgwAuditTransientErrorAbove10Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
        ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
        ingress traffic'
        expr:
(sum(rate(provgw_audit_transient_error{kubernetes_namespace="provgw"}
        [10m]))/
sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}[10m])))*100
>= 10 < 25
        labels:
          severity: Minor
      - alert: ProvgwAuditTransientErrorAbove25Percent
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(provgw_audit_transient_error{kubernetes_namespace="provgw"}
[10m]))
        /sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}
[10m])))*100 >= 25 < 50
        labels:
          severity: Major
      - alert: ProvgwAuditTransientErrorAbove50Percent
```

```
        annotations:
          description: 'Total number of response if subscriber not
found is about 50% of
          ingress traffic'
          summary: 'Total number of response if subscriber not found is
about 50% of
          ingress traffic'
        expr:
(sum(rate(provgw_audit_transient_error{kubernetes_namespace="provgw"}
[10m]))
        /sum(rate(provgw_audit_total{kubernetes_namespace="provgw"}
[10m])))*100 >= 50
        labels:
          severity: Critical
```

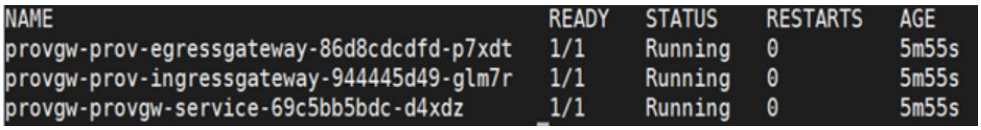
7
Troubleshooting Provisioning Gateway

You can avoid most of the troubleshooting issues, if Provisioning Gateway is installed properly. Follow the steps given below to verify its installation:

- Execute the following command to verify the working condition of Provisioning Gateway specific pods:
  `kubectl get pods -n <provw-namespace>`

  A sampe output screen is shown below:

Figure 7-1 Provisioning Gateway Pod Status



> **Note:**
>
> All the pods in the above screen are running. If not, refer to sections below on possible reasons for pod creation failure.

Checking PROVGW-SERVICE Logs

The **PROVGW-SERVICE** has all the response status from both the UDRs. You can check its logs to view the response from both the UDR's. Execute the following command to view the logs:

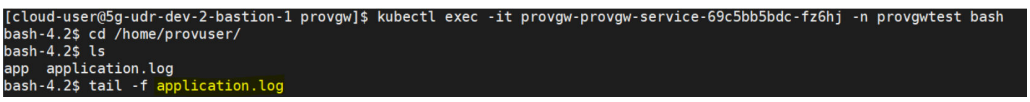
`kubectl logs <provgw-service pod> -n <provgw-namespace>`

Alternatively, you can check the logs directly on the pods using the following command:

`kubectl exec –it <provgw-service pod> -n <provgw-namespace> bash`

A sample screenshot is given below:

Figure 7-2 PROVGW-SERVICE Logs



You can check the logs in the **application.log** file.

Changing PROVGW-SERVICE Logging Level

> **Note:**
>
> You need to redeploy the setup for the changes to take effect.

To change the PROVGW-SERVICE logging level using Helm:

1. Open the latest provgw_value.yaml file used during Provisioning Gateway installation.
2. Under provgw-service, change the value of "logging level root" attribute to "INFO"/"DEBUG".

   Extract from provgw_values.yaml

   ```
   provgw-service:
   ...
   ...
   ...
       logging:
           level:
             root:"WARN"
   ```

   Other logging level values are DEBUG, INFO, WARN, ERROR.

Debugging Pod Creation Failure

A pod creation may fail due to any one of the following reasons:

- **Incorrect Pod Image:** You need to check if any pod is in the **ImagePullBackOff** state. If it is there, it means that the image name used for one of the pods is not correct. You need to check the following values in the values.yaml file.

  ```
  provgw-service:

         ...

           image:

                name: reg-1:5000/provgw/provgateway

                tag: 1.8.0

        prov-ingressgateway:

         ...

           image:

                name: reg-1:5000/provgw/ocingress_gateway

                tag: 1.8.0
  ```

```
prov-egressgateway:

 ...

   image:

        name: reg-1:5000/provgw/ocegress_gateway

        tag: 1.8.0
```

After updating the values.yaml file, execute the following command to install helm:

```
helm install –-name <release-name> --namespace <release-namespace>
```

It helps you to purge the old setup and reinstall for the changes or upgrade the helm instance.

- **Resource Allocation Failure:** You need to check if any pod is in the Pending state. If yes, then execute the following command for those pods:
  `kubectl describe <provgw-service pod id> --n <provgw-namespace>`

  In the output, check whether there is any warning for Insufficient CPU. If any warning is found, it means there are not sufficient CPU resources to start the pod.

  You can address this issue by either increasing the number of CPUs as a hardware or reducing the number of CPUs allotted to a pod in the values.yaml file.

```
provgw-service:
. . .
. . .
. . .
resources:
  limits:
    cpu: 3
    memory: 4Gi
  requests:
    cpu: 3
    memory: 4Gi
prov-ingressgateway:
...
...
...
resources:
  limits:
    cpu: 3
    memory: 4Gi
  requests:
    cpu: 3
    memory: 4Gi
```

After updating the values.yaml file, execute the following command to install helm:

```
helm install –-name <release-name> --namespace <release-namespace>
```

Debugging UDR Registration with ProvGw

Before deploying Provisioning Gateway, you need to ensure that UDR pods are in running state and the FQDN of UDRs are correct.

In the helm charts, UDR FQDNs information is available in below format:

Figure 7-3 UDR FQDN Info

```
segDetails:
  - name: SEG-1
    fqdnValues: udr1-ingressgateway.udr1,udr2-ingressgateway.udr2
    preferred: udr1-ingressgateway.udr1
  - name: SEG-2
    fqdnValues: udr3-ingressgateway.udr3,udr3-ingressgateway.udr4
    preferred: udr3-ingressgateway.udr3
```

Initially, Provisioning Gateway registers with the preferred FQDN in each segment irrespective of its active state and later, it internally verifies the status of other FQDNs in each segment and updates, if the preferred FQDN is down.

Before deploying Provisioning Gateway, you need to ensure that:

- All the pods of UDR are in running state.
- FQDN of the UDR is correctly mentioned.
- The preferred FQDN is considered as active when all the FQDN's are down in one segment.
- The active FQDN is checked every 15 seconds (This value is configurable).

In Provisioning Gateway (ProvGw),

- The get requests sent to UDR for updating the active FQDN, are not dumped in the logs of ProvGw.
- There is only one `provgw-service` that receives all the requests from Ingress Gateway. If any request fails:
  - **With 500 status code without Problem Details information:** It means the flow ended in **prov-ingressgateway service** pod without route. To confirm the same, you need to check the prov-ingressgateway pod logs for errors/ exception.
  - **With 503 status code with SERVICE_UNAVAILABLE in Problem Details:** It means the **provgw-service** pod is not reachable due to some reason.

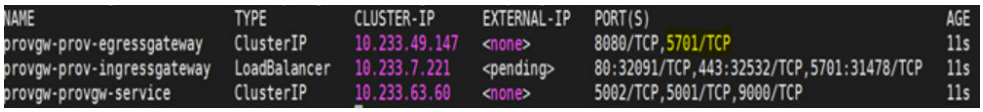
Figure 7-4 503 Status Code

You can confirm this error in the prov-ingressgateway pod logs for errors/ exception. Check the provgw-service pod status and fix the issue.

- Try to find the root cause using metrics as follows:
  - If the count of **oc_ingressgateway_http_requests_total** measurement increases, then check the content of incoming requests. You need to ensure that the incoming JSON data blob is as per specification.
  - If the **udr_rest_request** measurement increases more than one per request then you need to ensure that UDR is working fine and the Ingress Gateways of UDR are not down.

- To debug HTTPS related issues, refer to **Unified Data Repository Installation and Upgrade Guide**.

Figure 7-5 ProvGw - HTTPS Port Exposure



- To debug HPA Issues, refer to **Unified Data Repository Installation and Upgrade Guide**

Debugging Provisioning Gateway with Service Mesh Failures

There are some known failure scenarios that you may encounter while installing provisioning gateway 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 7-6 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 7-7 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.

- If the provisioning system outside service mesh is not able to contact provgw service through its ingress gateway, then you have to:

- Exclude the HTTP container port traffic for prov-ingressgateway from istio side car.
- Configure proper port as shown below under prov-ingressgateway section.

Figure 7-8 Annotation under prov-ingressgateway section

```
deployment:
  customExtension:
    labels: {}
    annotations:
      traffic.sidecar.istio.io/excludeInboundPorts: "\"8081\""
```

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

Figure 7-9 Annotation to View ProvGw 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: Custom Extensions Global Parameters ********
#************************************************************************
```

Troubleshooting provgw-service via Metrics

If provgw-service requests fail, you can try to find the root cause from metrics as well. Some of the troubleshooting tips are as follows:

- If the count of **oc_ingressgateway_http_requests_total** measurement increases, you should check the content of incoming request and make sure that incoming json data blob is proper and as per the specification.
- If on one request, the **udr_rest_request** measurement increases more than once then:

- Make sure the UDR's are working fine
- Make sure the ingress gateways of UDR are not down.

8
Upgrading an Existing ProvGateway Deployment

In this section, you will learn to upgrade an existing Provisioning Gateway Deployment.

> **Note:**
>
> **You can upgrade from ProvGateway 1.7.1 to ProvGateway 1.8.0**

Upgrading an existing deployment replaces the running containers and pods with new ones. If there is no change in the pod configuration, the pods are not replaced. Unless there is a change in the service configuration of a microservice, the service endpoints remain unchanged (ClusterIP etc.).

> **Note:**
>
> You should stop the provisioning traffic while upgrading the Provisioning Gateway and then, perform the helm upgrade.

Helm Upgrade
To upgrade Provisioning Gateway via Helm:

- Follow the deployment instructions provided in the Installation Sequence section for extracting the required Provisioning Gateway software components and if required, re-tag and push the images to customer's repository.
- Take a backup of the provgw-custom-values-1.8.0.yaml file and modify the parameters as per site requirement.
- Execute the following command to upgrade an existing Provisioning Gateway deployment. For the parameters that are configurable, see Customizing Provisioning Gateway.
  ```
  $ helm upgrade <release> <helm chart> [--version <Provgw version>] -f
  <provgw-custom-values-1.8.0.yaml>
  ```
  In the above command:
  - **<release>** information is available in the output of 'helm list' command.
  - **<chart>** is the name of the chart in the form of <repository/provgw> .
    Example: reg-1/provgw or cne-repo/provgw

Rollback Instructions
Execute the following command to check whether pods are running successfully:

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

If there are issues that you cannot recover on checking logs and describe on pods, then you have to do image rollback using Helm. The steps are as follows:

- Use the backed up customized previous version's <provgw-values.yaml> file to roll back to the previous version.
- Execute the helm rollback command, `helm rollback <helm release name> <revision_no>`.

> **Note:**
>
> To obtain the revision number, execute this command: `helm history <helm release name>`

9
Uninstalling Provisioning Gateway

In this section, you will learn to uninstall Provisioning Gateway.

To uninstall or completely delete the provisioning gateway deployment, execute the following command:

```
helm del --purge <helm_release_name_for_provgw>
```

> **Note:**
>
> If helm3 is used, then execute the following command:
> `helm uninstall <helm_release_name_for_provgw> --namespace <provgw_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
```