Oracle® Communications

Cloud Native Core, Network Function Data Collector User Guide

Release 1.2.5
F79383-05
December 2023

Oracle Communications Cloud Native Core, Network Function Data Collector User Guide, Release 1.2.5

F79383-05

Contents

My Oracle Support

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

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

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

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

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

Acronyms

The following table provides information about the acronyms and the terminology used in the document:

Table Acronyms

Acronym	Description
CNC	Cloud Native Core
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
FQDN	Fully Qualified Domain Name
Gi	Gigabytes
K8s	Kubernetes
Kubectl	Kubernetes
Mi	Megabytes
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
OSDC	Oracle Software Delivery Cloud
PCF	Policy Control Function
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
UTC	Universal Time Coordinated

What's New in This Guide

This section introduces the documentation updates for release 1.2.5.

Release 1.2.5 - F79383-05, December 2023

- Updated Step 2 with new parameters to enable the collection of requested Kubernetes resources, resources pertaining only to a requested microservice, and to collect previous pod logs. Also, added Example 7 with sample output.

Release 1.2.4 - F79383-04, August 2023

- Added examples of the Kubernetes resources in the Network Function Deployment Data Collector section.
- Updated Step 2 with sample output and a new command to determine the version of NF Data Collector.

Release 1.2.3 - F79383-03, July 2023

Added a note about using the `-f|--nfLogStatus =[true]` element at Step 2 and added a descrioption for `[NF PODS LOG COLLECTION STATUS]` in the Using the NF Deployment Data Collector Module section.

Release 1.2.3- F79383-02, April 2023

Updated the release version of CNC NF Data Collector to 1.2.3 throughout the document.

Release 1.2.2- F79383-01, March 2023

- Added the following tables in the Network Function Deployment Data Collector section to collect cnDBTier logs from these tables:
  - REPLICATION_CONNECTION_STATUS
  - NDB_BINLOG_INDEX
- Updated the command and the step result about tarball division in Step 2.

1

Introduction

This document contains information about the functionalities of Cloud Native Core (CNC) Network Function (NF) Data Collector. Using this document, you can perform the following tasks:

- Download and prepare CNC NF Data Collector modules.
- Run the Data Collector to generate tarballs that can be used to debug any NF deployment issue.
- Export and load the NF data such as logs, alarms, metrics, and traces to another system.

2 Acronyms

The following table provides information about the acronyms and the terminology used in the document:

Table 2-1 Acronyms

Acronym	Description
CNC	Cloud Native Core
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
FQDN	Fully Qualified Domain Name
Gi	Gigabytes
K8s	Kubernetes
Kubectl	Kubernetes
Mi	Megabytes
NF	Network Function
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
OSDC	Oracle Software Delivery Cloud
PCF	Policy Control Function
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
UTC	Universal Time Coordinated

3
Understanding CNC NF Data Collector

CNC NF Data Collector collects Network Functions' deployment data when NFs are deployed. Also, it collects logs, metrics, traces, and alerts when any functionality issue occurs after the NF deployment is complete.
You can run CNC NF Data Collector to generate the required tarballs and examine the log files present in the tarballs for debugging purposes. Using this data, you can determine an NF issue by accessing logs, metrics, traces, and alerts of the NF.

CNC NF Data Collector consists of the following modules:

- Network Function Deployment Data Collector
- Network Function Logs, Metrics, Traces, Alerts Data Collector

3.1 Network Function Deployment Data Collector

This module collects NFs deployment data when deploying or upgrading the NFs and when any node is down or inactive. This module collects the following NF specific deployment data:

- Helm deployment status mapped to Namespace entered by users.
- Helm configuration used for deploying the NF mapped to Namespace entered by users.
- Status of all the Kubernetes resources used in the Helm deployment.
- Deployment description of all the Kubernetes resources, such as pods, deployments(deploy), services(svc), replicasets(rs), horizontalpodautoscaler(hpa), configmaps(cm), PodDisruptionBudeget(pdb), EndPoint(ep), used in the Helm deployment.
- Logs from all the pods in the Helm deployment.
- Kubernetes events.
- cnDBTier logs.
  If the cnDBTier log collection is enabled through the `cndbTierLogCollection` parameter, it collects data from DBTIER_REPLICATION_CHANNEL_INFO, DBTIER_REPL_SITE_INFO, DBTIER_INITIAL_BINLOG_POSTION, REPLICATION_CONNECTION_STATUS, and NDB_BINLOG_INDEX tables. Also, it collects the following data:
  - Slave status, bin logs, relay, and MySQL logs from MySQL pods.
  - Trace logs and cluster status from MySQL data pods.
  - Cluster logs from MySQL management pods.

> **Note**
>
> If the Helm release is unavailable, then Helm specific data is not collected. If multiple Helm releases are mapped to the same Kubernetes namespace, then Helm specific data is collected for all Helm releases mapped to entered namespace.

3.2 Network Function Logs, Metrics, Traces, Alerts Data Collector

This module collects NF specific data such as logs, metrics, traces, and alerts, when a deployed NF malfunctions or fails. Using this module, you can collect the required log files to debug NF related issues. This module provides the following utilities:

- Exporter utility
- Loader utility

Exporter Utility

Run this utility on the same system from which the data has to be collected. Exporter utility uses Elastic Search and Prometheus Server to collect the following data for a specified time interval provided in the `exporter-custom-values.yaml` file:

- Logs
- Metrics
- Traces
- Alarms

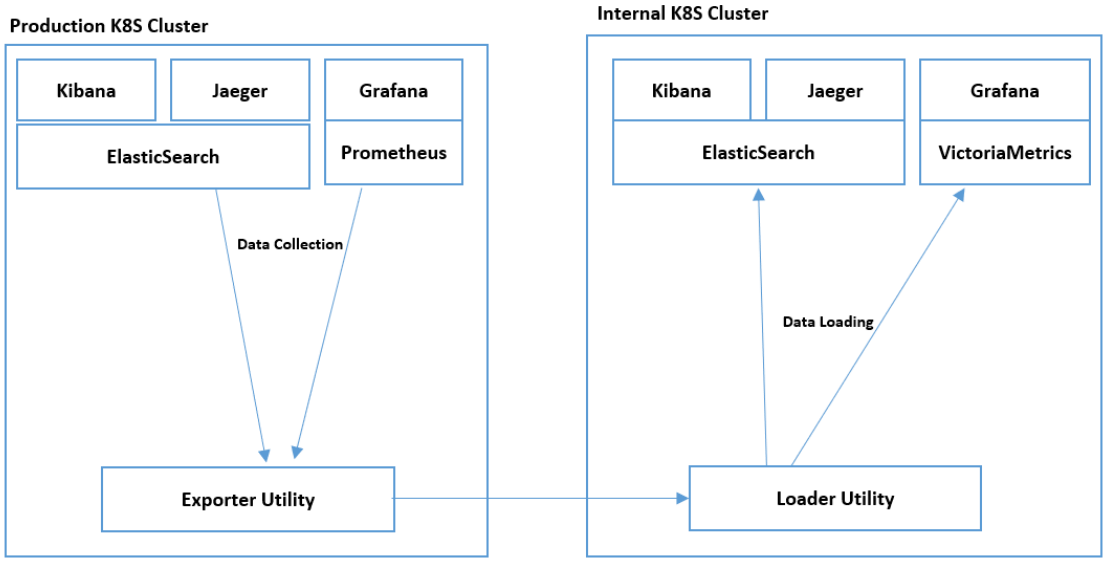
Loader Utility

Run this utility on the same system to which the data needs to be loaded in plain files. Loader utility uses Elastic Search and Victoria Metrics to load the same data that was exported using the Exporter utility.

> **Note**
>
> Victoria Metrics is a time series database similar to Prometheus. Victoria Metrics is used instead of Prometheus because Prometheus does not support push mechanism to store data into its time series database.

The following image explains data collection and data loading by using Exporter and Loader utilities.

Figure 3-1 Data Collection and Data Loading



4
Preparing CNC NF Data Collector Modules

This section describes the prerequisites and download procedure for CNC NF Data Collector.

4.1 Prerequisites

This section includes information about the required prerequisites before downloading CNC NF Data Collector:

- Kubernetes 1.13 or later, Helm 2, and Helm 3 must be installed on the host machine.
- This tool must be run on a system or machine (Bastion Host) from where the NF is deployed using Helm or Kubectl.
- Logs must be stored in Elastic Search, Metrics, and Alarms in Prometheus Server. Jaeger must be configured to use Elastic Search as jaeger's storage database. All versions of the third-party tools are supported.

4.2 Downloading and Preparing CNC NF Data Collector

Perform this procedure to download Cloud Native Core (CNC) Network Function (NF) Data Collector Images and Helm files from Oracle Software Delivery Cloud (OSDC) and prepare CNC NF Data Collector to start.

1. Download the following files from OSDC:
   - CNC NF Data Collector package file: `<name>-pkg-<marketing-release-number>.tgz`
     Example:

     ```
     ocnfDataCollector-pkg-1.2.5.0.0.tgz
     ```

   - ReadMe file: `<name>-pkg-<marketing-release-number>-README.txt`
     Example:

     ```
     ocnfDataCollector-pkg-1.2.5.0.0-README.txt
     ```

2. Run the following command to untar the CNC NF Data Collector package:

   ```
   tar -xvf <<name>-pkg-<marketing-release-number>>.tgz
   ```

   The system creates the `<<nfname>-pkg-<marketing-release-number>>` directory, which contains the following items:

   ```
   ocnfDataCollector-pkg-1.2.5.0.0.tgz
              |_ _ _ _ _ _ ocnfDeploymentDataCollector-1.2.5.tgz
              |_ _ _ _ _ _ ocnf-data-exporter-1.2.5.tgz (Helm Charts)
              |_ _ _ _ _ _ ocnf-data-loader-1.2.5.tgz (Helm Charts)
              |_ _ _ _ _ _ ocnfDataCollector-image-1.2.5.tar (Docker Images)
   ```

```
|_ _ _ _ _ _ exporter-custom-values.yaml
|_ _ _ _ _ _ loader-custom-values.yaml
```

3. Verify the checksum of packages from the `<name>-pkg-<marketing-release-number>-README.txt` file.

   ```
   cksum ocnfDataCollector-pkg-1.2.5.0.0-README.txt


   2932245484 2256 ocnfDataCollector-pkg-1.2.5.0.0-README.txt
   ```

4. Run the following command to load the images to the local registry:

   ```
   docker load --input ocnfDataCollector-image-1.2.5.tar
   ```

5. Run the `docker images` command to check whether all the images are loaded.
6. Run the following commands to push the docker images to docker registry:

   ```
   docker tag <image-name>:<image-tag> <docker-repo>/<image-name>:<image-tag>


   docker push <docker-repo>/<image-name>:<image-tag>


   docker tag cnc-nfdata-collector:1.2.5 kingbastion-2:5000/cnc-nfdata-collector:1.2.5
   ```

7. Run the following commands to untar the helm files:
   - `tar -xvzf <file name>`: To untar the `ocnf-data-exporter-1.2.5.tgz` file if you want to collect the data on the required system.
     Where, `<file name>` indicates the file name, for example, `tar -xvzf ocnf-data-exporter-1.2.5.tgz`.

     The system creates a directory with the name `ocnf-data-exporter`. This directory contains the following files: `Chart.yaml`, `nf-tools`, `templates`, and `values.yaml`.
   - `tar -xvzf <file name>`: To untar the `ocnf-data-loader-1.2.5.tgz` file if you want to import the data into the required system.
     Where, `<file name>` indicates the file name, for example, `tar -xvzf ocnf-data-loader-1.2.5.tgz`.

     The system creates a directory with the name `ocnf-data-loader`. This directory contains the following files: `Chart.yaml`, `templates`, and `values.yaml`.

5
Collecting Data Using CNC NF Data Collector

The following procedures provide information about how to start both the data collector modules and generate the output.

5.1 Using the NF Deployment Data Collector Module

Perform this procedure to start the NF Deployment Data Collector module and generate the tarballs. If the user does not specify the output storage path, then this module generates the output in the same directory where the module is running.

- Ensure that you have privileges to access the system and run kubectl and Helm commands.
- Prepare CNC NF Data Collector as described in Downloading and Preparing CNC NF Data Collector.
- Perform this procedure on the same machine (Bastion Host) where the NF is deployed using helm or kubectl.

1. To extract `ocnfDeploymentDataCollector-1.2.5.tgz`, run the following command:

   ```
   tar -xvf ocnfDeploymentDataCollector-1.2.5.tgz
   ```

   The data is extracted into the `ocnfDeploymentDataCollector` directory.

   - Run the `chmod +x nfDataCapture.sh` command on the tool to provide the executable permission.

2. Run the following command to start the module:

   Users with privileges to run kubectl and Helm commands can run this module. In case the context file is available with required privileges, then it can be used in Example 1.

   ```
   ./nfDataCapture.sh -v|--version - This will be used to check current
   running version of NF Data Collector on system.
     ./nfDataCapture.sh -n|--k8Namespace=[K8 Namespace] -u|--username=[User
   Name]
     -p|--password=[Password] -k|--kubectl=[KUBE_SCRIPT_NAME] -h|--
   helm=[HELM_SCRIPT_NAME]
     -d|--cndbTierLogCollection=[true/false] -b|--
   cndbTierbinlogCollection=[true/false] -f|--nfLogStatus=[NF PODS LOG
   COLLECTION STATUS]
     -a|--all=[COLLECT_ALL_DBTIER_BINLOG_NFLOG] -s|--
   size=[SIZE_OF_EACH_PART_TARBALL] -o|--toolOutputPath -helm3=false -rn|--
   resourceNames=[VALID k8 RESOURCE NAMES] -ms|--microService=[VALID
   MICROSERVICE NAME] -previous|--previousPodLogs=[true/false]
   ```

   Where,

   - `version` indicates the current version of Network Function Data Collector running on your system.

- `[K8 Namespace]` indicates Kubernetes Namespace in which NF is deployed.
- `[KUBE_SCRIPT_NAME]` indicates the kube script name.
- `[HELM_SCRIPT_NAME]` indicates the Helm script name.
- `[SIZE_OF_EACH_TARBALL]` indicates the size of each tarball.
- `[NF PODS LOG COLLECTION STATUS]` indicates the NF pod log collection status.
- `[VALID k8 RESOURCE NAMES]` indicates all the Kubernetes resource names, such as pods, svc, cm, pdb, and so on.
- `[VALID MICROSERVICE NAME]` indicates the microservice name “ingress-gateway” or “egress-gateway”.
- `-previous` enables the collection of previous pod logs when set to `true`.

> **Note**
>
> `-f|--nfLogStatus =[true]` must be used to collect NF pod logs.

This module generate tarball that starts with `<K8Namespace>debugData`, which contains all the logs required for debugging, and divides the same tar into several tarballs based on the size specified in the `-s` attribute. For more information, see Example 1 and Example 2. By default, if no size is specified, then the tarball does not get divided. For more information, see Example 3.
Examples of the command:

- **Example 1**:

  ```
  ./nfDataCapture.sh -k ="kubectl --kubeconfig= admin.conf" -h ="helm --
  kubeconfig admin.conf" -n=ocnrf -s=5M -o=/tmp/
  ```

- **Example 2**:

  ```
  ./nfDataCapture.sh -n=observability -s=5M -o=/tmp/ -f=true
  ```

  Values used:

  ```
  Kubectl command: kubectl
  Helm command: helm
  K8 Namespace: observability
  Bin Log Collection Status:  false
  CNDB Tier Log Collection Status:  false
  NF Log Collection Status:  true
  All log/bin Status:  false
  Output directory path: /tmp/
  Helm3: false
  Maximum size of each tarball: 5M
  ```

  Sample output:

  ```
  drwxrwxr-x. 3 opc  opc    8192 Aug  8 13:55
  observability.debugData.2023.08.08_13.51.50
  ```

```
-rw-rw-r--. 1 opc  opc  210238 Aug  8 13:55
observability.debugData.2023.08.08_13.51.50-dc_nflog.tar.gz
```

- **Example 3**:

  ```
  ./nfDataCapture.sh -n=observability -helm3=true -f=true
  ```

  Values used:

  ```
  Kubectl command: kubectl
  Helm command: helm3
  K8 Namespace: observability
  Bin Log Collection Status:  false
  CNDB Tier Log Collection Status:  false
  NF Log Collection Status:  true
  All log/bin Status:  false
  Output directory path: .
  Helm3: true
  ```

  Sample output when Helm3 is used:

  ```
  drwxrwxr-x. 3 opc  opc    8192 Aug  8 13:55
  observability.debugData.2023.08.08_13.51.50
  -rw-rw-r--. 1 opc  opc  210238 Aug  8 13:55
  observability.debugData.2023.08.08_13.51.50-dc_nflog.tar.gz
  ```

  Sample output when Helm3 is not used:

  ```
  helm3 does not exist or could not be found
  ```

- **Example 4**

  ```
  ./nfDataCapture.sh -n=observability -s=5M -o=/tmp/ -u=dbusername -
  p=dbpassword -d=true -b=false -helm3=false
  ```

  Values used:

  ```
  Kubectl command: kubectl
  Helm command: helm
  K8 Namespace: observability
  Bin Log Collection Status:  false
  CNDB Tier Log Collection Status:  true
  NF Log Collection Status:  false
  All log/bin Status:  false
  Output directory path: /tmp/
  Helm3: false
  Maximum size of each tarball: 5M
  ```

  Sample output:

  ```
  -rw-rw-r--. 1 opc  opc   404 Aug 10 17:14
  observability.debugData.2023.08.10_17.14.18-dc_dbtier.tar.gz
  ```

```
drwxrwxr-x. 3 opc  opc    23 Aug 10 17:14
observability.debugData.2023.08.10_17.14.18
```

- **Example 5**

  ```
  ./nfDataCapture.sh -n=dbtier1 -o=/tmp/ -u=dbusername -p=dbpassword -
  d=true -f=true
  ```

  Values used:

  ```
  Kubectl command: kubectl
  Helm command: helm
  K8 Namespace: dbtier1
  Bin Log Collection Status:  false
  CNDB Tier Log Collection Status:  true
  NF Log Collection Status:  true
  All log/bin Status:  false
  Output directory path: /tmp/
  Helm3: false
  ```

  Sample output:

  ```
  -rw-rw-r--. 1 opc  opc   95K Aug 10 17:23
  dbtier1.debugData.2023.08.10_17.19.41-dc_nflog.tar.gz
  -rw-rw-r--. 1 opc  opc  1.7M Aug 10 17:24
  dbtier1.debugData.2023.08.10_17.19.41-dc_dbtier.tar.gz
  drwxrwxr-x. 4 opc  opc  4.0K Aug 10 17:24
  dbtier1.debugData.2023.08.10_17.19.41
  ```

- **Example 6**

  ```
  ./nfDataCapture.sh -n=dbtier1 -a=true -u=dbusername -p=dbpassword
  ```

  Values used:

  ```
  Kubectl command: kubectl
  Helm command: helm
  K8 Namespace: dbtier1
  Bin Log Collection Status:  false
  CNDB Tier Log Collection Status:  false
  NF Log Collection Status:  false
  All log/bin Status:  true
  Output directory path: .
  Helm3: false
  ```

  Sample output:

  ```
  -rw-rw-r--. 1 opc opc 129K Aug  8 14:50
  dbtier1.debugData.2023.08.08_14.47.07-dc_nflog.tar.gz
  -rw-rw-r--. 1 opc opc 384K Aug  8 14:52
  dbtier1.debugData.2023.08.08_14.47.07-dc_dbtier.tar.gz
  -rw-rw-r--. 1 opc opc 1.8M Aug  8 14:52
  dbtier1.debugData.2023.08.08_14.47.07-dc_binlog.tar.gz
  ```

- **Example 7**

```
./nfDataCapture.sh -n=apigw -u=root -p=NextGen -s=5M -o=/root/
datacollector/ -helm3=false -d=false -b=false -rn=po,svc,cm -ms=ingress-
gateway -previous=true -f=true
```

Values used:

```
Kubectl command: kubectl
Helm command: helm
K8 Namespace: apigw
Bin Log Collection Status:  false
CNDB Tier Log Collection Status:  false
NF Log Collection Status:  true
All log/bin Status:  false
Output directory path: /root/datacollector
Resource Names: po,svc,cm
Microservices: ingress-gateway
Previous Pod Logs Collection: true
Helm3: false
Maximum size of each tarball: 5M
```

Sample output:

```
apigw.debugData.2023.12.19_07.35.55-dc_nflog.tar.gz
```

> **Note**
>
> Tarball does not get created in parts if the `-s` attribute is not passed with a specific size. The default location of the output is tool working directory if the location is not specified. By default, helm2 is used. Use proper argument in command to use helm3.

If the size of the tar, for example, ocnrf.debugData.2020.06.08_12.02.39.tar.gz, generated is greater than "SIZE_OF_EACH_TARBALL" specified in the command, then the tar is split into several tarballs based on the size specified.

> **Note**
>
> - The `cndbTierLogCollection` parameter is optional. It can be enabled by setting `"-d=true"`, along with inputs for the dbusername and dbpassword. `cndbTierbinlogCollection` collects the bin logs from mysqld pod. `nfLogStatus` is optional and is enabled by setting `"-f=true"`. It collects logs from all the pods. An option to collect all of the above mentioned logs is provided by setting `"-a=true"`, along with inputs for the dbusername and dbpassword.
> - Ensure that the aforementioned script has the permissions of execution in the environment. `chmod +x nfDataCapture.sh` can be used in to grant the permissions. Ensure that the tar package and split package are installed in the environment.

Sample output

```
NFDataCollectorVersion: 1.2.5

Executing:  ./nfDataCapture.sh -k="kubectl --kubeconfig=admin.conf" -
h="helm --kubeconfig=admin.conf" -n=ocnrf -s=5M -o=/tmp/
2023.08.24_09.57.06 :: Data capture tool execution started

---------------------------------------

Data capture tool logs are available at ./
nfDataCaptureToolLogs.2023.08.24_09.57.05.log

---------------------------------------

Data collection work completed. Packing data collected ....
Data collected to :- ./ocnrf.debugData.2023.08.24_08.30.19.tar.gz
```

3. After transferring the tarballs to the required destination, combine the tarballs into a single tarball by running the following command:

> **Note**
>
> If the size of the generated tar, for example, `ocnrf.debugData.2020.06.08_12.02.39.tar.gz`, is greater than the size specified in the command, then the tar is split into several tarballs based on the size specified in `<SIZE_OF_EACH_TARBALL>`.

```
cat <splitted files*>  <combinedTarBall>.tar.gz
```

Example:

```
cat ocscp.debugData.2020.09.14_18.12.49-part* >
ocscp.debugData.2020.09.14_18.12.49-combined.tar.gz
```

5.2 Start the NF Logs, Metrics, Traces, Alerts Data Collector Module

The following table outlines the sequence of tasks to be performed to start the module and view the collected data.

Table 5-1 Start the NF Logs, Metrics, Traces, Alerts Data Collector Module

Task Sequence	Task Name	Description	Section Reference
1	Pre-deployment task	Use this task to remove the residual or remaining resources of the previous deployment.	Pre-deployment Tasks of Logs, Metrics, Traces, Alerts Data Collector Module

Table 5-1 (Cont.) Start the NF Logs, Metrics, Traces, Alerts Data Collector Module

Task Sequence	Task Name	Description	Section Reference
2	Configuration of Exporter utility	Use this task to configure the Exporter utility and to customize the `exporter-custom-values.yaml` file.	Creating or Updating the YAML File for the Exporter Utility
3	Starting of Exporter utility	Use this task to start the Exporter utility on the same site from which the data needs to be collected.	Exporting the NF Logs, Metrics, Traces, and Alerts Data
4	Check the output of Exporter utility	Use this task to verify the output of the Exporter utility.	Sample Content of the Exporter Utility Custom values.yaml
5	Transfer the data collected by Exporter utility	Use this task to transfer the data collected by the Exporter utility.	Transferring the Data Collected by Exporter Utility
6	Configuration of Loader utility	Use this task to configure the Loader utility and to customize the `loader-custom-values.yaml` file.	Creating or Updating the YAML File for the Loader Utility
7	Starting of Loader utility	Use this task to start the Loader utility on the data collected from the site.	Loading the NF Logs, Metrics, Traces, Alerts Data
8	View NF Logs, Metrics, and Traces	Use this task to view the data collected from the site.	Viewing the NF Logs, Metrics, and Traces
9	Delete the logs, metrics, traces data	Use this task to delete the data after the analysis is complete. **Note**: This is an optional task.	Deleting the NF Logs, Traces, and Metrics

5.2.1.1 Removing the Pending Resources from Namespace

It is observed that some resources of the Network Function Logs, Metrics, Traces, and Alerts Data Collector module are not deleted while removing the deployment. Perform the following procedure to remove pending resources from namespace.

1. Run the following command to check the presence of pending resources:

```
$ kubectl get all -n <tool-namespace>


kubectl get all -n ocnf-data-loader

NAME                                      READY   UP-TO-DATE   AVAILABLE
AGE
ocnf-data-loader-dsgsf643tr           1/1     1            1
4d21h
```

The system retrieves a detailed overview of the current objects of `<tool-namespace>`.

2. Run the following command to delete pending resources from namespace:

   ```
   $ kubectl delete <resource-type> <resource-name> -n <tool-namespace>
   ```

   And we need to delete the deployment of ocnf-data-loader kubectl delete deployment ocnf-data-loader -n ocnrf deployment.extensions "ocnf-data-loader" deleted

3. **⚠ Caution**

   The command in this step removes all the resources.

   Run the following command to clean up all the pending resources and remove all the Kubernetes objects:

   ```
   kubectl delete all --all -n <tool-namespace>
   ```

5.2.2 Creating or Updating the YAML File for the Exporter Utility

The Export utility YAML file is present in the CNC NF Data Collector tar package. You can either update the existing YAML file or create a new YAML file with the following parameters which can be configured. Oracle recommends to consider the existing file that is available with the tar package and update the parameters.
For information about parameter descriptions, refer to exporter-custom-values.yaml Parameter Description.

1. In the `exporter-custom-values.yaml` file, update the `global.image.repository` parameter by providing the name of the docker registry where the `cnc-nfdata-collector` image is present.
2. Update the `global.slaveNodeName` parameter by providing the name of the kubernetes worker node that can store the collected data from this utility.

   To obtain the name of the worker node or slave node, run the `kubectl get nodes` command. The names of all the master and worker nodes of the kubernetes cluster are displayed in the Name column. You can provide the name of one of the worker nodes from the generated output.
3. Update the `global.outputPath` parameter by providing the path to collect the data on the kubernetes cluster worker node in `global.slaveNodeName`.

   - Ensure that the path provided here already exists on the kubernetes worker node specified in `global.slaveNodeName` and is accessible by non-root users for read or write operation. The `/tmp` path is recommended because it qualifies the required behavior on most of the kubernetes cluster nodes.

   The Exporter utility creates the `exported-data_<time-stamp>/` directory in this path.
4. Update the `global.capacityStorage` parameter by specifying the estimated amount of space required to occupy the collected data, for example, 2Gi, 200Mi, and so on.

   The value specified here is the space provided to kubernetes persistence volume that is mounted with this utility to collect the data.
5. Update the `global.storagePod` parameter.

> **Note**
>
> If the user does not have access to worker node, where the data is collected, then the value of this parameter must be set to `true`. Otherwise, the value is `false` if the user has access to worker node. The system creates a storage pod and mounts it with the persistence volume that collects the data at the `/volume` path in the pod.

- To copy data from the storage pod, run the following kubectl cp command:

  ```
  kubectl cp exporter-storage-pod:/volume/exported-data_<time-stamp> ./ -
  n <namespace>
  ```

  Example:

  ```
  kubectl cp exporter-storage-pod:/volume/exported-
  data_2020-08-24_09:40:52 ./ -n ocnf-data-exporter
  ```

6. Update the `global.elasticSearchURL` parameter by specifying the URL for Elastic search.

   FQDN of Elastic search can also be provided. If Elastic search requires authentication, it can be provided in the URL as `http://<user-name>:<password>@<elastic-search-url>:<elastic-search-port>`.
   To find the FQDN, run `kubectl get svc –n <namespace>` and retrieve the Elastic search service name. After that, FQDN can be constructed as `http://<elastic-search-service-name>.<namespace>:<port>`. The default Elastic search port is 9200. The administrator of the cluster must be asked for the user name and password of Elastic search if required.

   > **Note**
   >
   > Use the elasticsearch-client FQDN for Oracle Communications Cloud Native Environment (OCCNE).

7. Update the `global.prometheusURL` parameter by specifying the URL for the Prometheus server.

   FQDN of Elastic search can also be provided. The Prometheus server URL can be provided as `http://<prometheus-server-url>:<prometheus-server-port>`.
   To find the FQDN, run `kubectl get svc –n <namespace>` and retrieve the Prometheus server service name. After that, FQDN can be constructed as `http://<prometheus-server-service-name>.<namespace>:<port>`. The default Prometheus server port is 80.

   The namespace is not provided if the Prometheus server and the Exporter utility are in the same namespace.

> **Note**
>
> There is a separate configuration known as prefix path for Prometheus URL. If any prefix path is present, configure `prometheusPrefixPath: "<Prometheus Prefix Path>"`, for example, `prometheusPrefixPath: "/jazz/prometheus"`. Otherwise, leave the `prometheusPrefixPath:` field blank.

8. Update or remove the `LogsExporter.inclusionFilters` parameter as required.

   You must provide this parameter if you want to filter logs data. If you want to collect all the data, do not provide this parameter. Inclusion filters accept an array of filters. Each array element is accepted as OR rule. Within each array element, filters separated by comma (,) are taken as AND rule. Where, AND and OR are logical operators.

   ```
   LogsExporter.inclusionFilters:
     - node=worker-2,namespace=nssf
     - node=worker-2,namespace=pcf
   ```

   The aforementioned filter is interpreted as:

   ```
   (node=worker-2 AND namespace=nssf) OR (node=worker-2 AND namespace=pcf)
   ```

   Inclusion filter examples are described in Examples of Logs, Metrics, and Traces Inclusion Filters.

   For information about logs inclusion filter keywords, refer to NF Specific Logs Inclusion Filter Keywords.

   > **Note**
   >
   > Log filters are applied to JSON fields.

9. Update or remove the `LogsExporter.exclusionFilters` parameter as required.

   You must provide this parameter only if you want to exclude some logs data. Exclusion filters accept an array of filters. Each array element is accepted as OR rule. Within each array element, filters separated by comma (,) are taken as AND rule as described in Step 8. The only difference is that the logs applicable for the rules are excluded. AND and OR are logical operators.

10. Update the `global.interval` parameter by specifying the interval in hours for which the data is required.

    The collected data will be of last interval hour from now. For example, if the interval is set to 1, the data collector collects data for the last one hour.

11. Update or remove the `global.pastTimeSpot` parameter as required.

    This parameter along with interval parameter specifies the time range for which the data has to be collected.
    This parameter accepts time in the UTC format. By default, the Exporter utility collects the last one hour data. This parameter can be used to override this default behavior. For example, if you want to generate the data from `2020-05-17T15:30:38Z` to `2020-05-17T17:30:38Z`, then you must set `pastTimeSpot` to `2020-05-17T17:30:38Z` and interval to `2`. The system considers the current time to be `2020-05-17T17:30:38Z` and goes

back 2 hours in time to collect the data ranging from `2020-05-17T15:30:38Z` to `2020-05-17T17:30:38Z`.

12. Update or remove the `MetricsExporter.inclusionFilters` parameter.

    You must provide this parameter if you want to filter metrics data. If you want to collect all the metric data, do not provide this parameter. Metrics Inclusion filters accept an array of filters. Each array element is accepted as OR rule. Within each array element, filters separated by comma (,) are taken as AND rule. Where, AND and OR are logical operators.

    ```
    MetricsExporter.inclusionFilters:
      - node=worker-2,namespace=nssf
      - node=worker-2,namespace=pcf
    ```

    The aforementioned filter is interpreted as:

    ```
    (node=worker-2 AND namespace=nssf) OR (node=worker-2 AND namespace=pcf)
    ```

    Inclusion filter examples are described in Examples of Logs, Metrics, and Traces Inclusion Filters.

    For information about metrics inclusion filter keywords, refer to NF Specific Metrics Inclusion Filter Keywords.

    > **Note**
    >
    > Metrics filters are applied to Metrics labels.

13. Update or remove the `TracesExporter.inclusionFilters` parameter as required.

    You must provide this parameter if you want to filter traces data. If you want to collect all the traces data, do not provide this parameter. Traces Inclusion filters accept an array of filters. Each array element is accepted as OR rule. Within each array element, filters separated by comma (,) are taken as AND rule. Where, AND and OR are logical operators.

    ```
    TracesExporter.inclusionFilters:
      - node=worker-2,namespace=nssf
      - node=worker-2,namespace=pcf
    ```

    The aforementioned filter is interpreted as:

    ```
    (node=worker-2 AND namespace=nssf) OR (node=worker-2 AND namespace=pcf)
    ```

    Inclusion filter examples are described in Examples of Logs, Metrics, and Traces Inclusion Filters.

    > **Note**
    >
    > Trace filters are applied to process tags.

14. Update or remove the `TracesExporter.exclusionFilters` parameter.

    You must provide this parameter only if you want to exclude some traces data.

Exclusion filters accepts an array of filters. Each array element is accepted as OR rule. Within each array element, filters separated by comma (,) are taken as AND rule. Where, AND and OR are logical operators. The only difference is that the traces applicable for the rules are excluded.

5.2.2.1 Examples of Logs, Metrics, and Traces Inclusion Filters

The following examples describe how to use inclusion filters in different scenarios:

Examples of Logs Inclusion Filters

Scenario 1

JSON Logs:

```
{“node”:”worker-1”}
{“node”:”worker-2”}
```

For aforementioned logs, if you want to collect logs of only worker-2 node, then you must use the inclusion filter as follows:

```
LogsExporter.inclusionFilters:
  - node=worker-2
```

Scenario 2

JSON Logs:

```
{“node”:”worker-1”,”namespace”:”nssf”}
{“node”:”worker-2”,”namespace”:”nssf”}
{“node”:”worker-2”,”namespace”:”pcf”}
```

For aforementioned logs, if you want to collect logs of worker-2 node and Network Slice Selection Function (NSSF) namespace, then you must use the inclusion filter as follows:

```
LogsExporter.inclusionFilters:
  - node=worker-2,namespace=nssf
```

Scenario 3

JSON Logs:

```
{“node”:”worker-1”,”namespace”:”nssf”}
{“node”:”worker-2”,”namespace”:”nssf”}
{“node”:”worker-2”,”namespace”:”pcf”}
{“node”:”worker-2”,”namespace”:”nrf”}
```

For aforementioned logs, if you want to collect logs of worker-2 node in Network Slice Selection Function (NSSF) and Policy Control Function (PCF) namespace, then you must use the inclusion filter as follows:

```
LogsExporter.inclusionFilters:
  - node=worker-2,namespace=nssf
  - node=worker-2,namespace=pcf
```

Examples of Metrics Inclusion Filters

Scenario 1

Metrics:

```
http_request{“node”:”worker-1”}
http_request{“node”:”worker-2”}
```

For aforementioned metrics, if you want to collect metrics of only worker-2 node, then you must use the inclusion filter as follows:

```
MetricsExporter.inclusionFilters:
  - node=worker-2
```

Examples of Traces Inclusion Filters

Scenario 1

Traces:

```
{ "traceID": "75e9080f1ab45425", "spanID": "75e9080f1ab45425",
"operationName": "10.178.246.56:32333", "startTime": 1582020558473011,
"startTimeMillis": 1582020558473, "duration": 14763, "process":
{ "serviceName": "ocnssf-nsgateway-ocnssf","tags": [{"key": "node", "type":
"string", "value": "worker-2"}]}}
{ "traceID": "75e9080f1ab45425", "spanID": "75e9080f1ab45425",
"operationName": "10.178.246.56:32333", "startTime": 1582020558473011,
"startTimeMillis": 1582020558473, "duration": 14763, "process":
{ "serviceName": "ocnssf-nsgateway-ocnssf","tags": [{"key": "node", "type":
"string", "value": "worker-1"}]}}
```

For aforementioned traces, if you want to collect traces of only worker-2 node, then you must use the inclusion filter as follows:

```
TracesExporter.inclusionFilters:
  - node=worker-2
```

5.2.2.2 exporter-custom-values.yaml Parameter Description

The following table describes the parameters which can be customized while updating the `exporter-custom-values.yaml` file.

Table 5-2 exporter-custom-values.yaml Parameters

Parameter	Description	Default Value	Range or Possible Value
`global.image.repository`	Specifies the name of the docker registry that contains the cnc-nfdata-collector image. Provide the name of the docker registry where the cnc-nfdata-collector image is present.	-	-
`global.slaveNodeName`	Specifies the name of the Kubernetes slave that stores the collected data. To obtain the name of the slave node or worker node, run the `kubectl get nodes` command. Then, provide the name of one of the worker nodes as mentioned in the Name column of the generated output. Provide the name of the Kubernetes slave where collected data by the tool is stored.	-	-
`global.outputPath`	Creates the `exported-data/` directory in the specified path that can be copied from the slave node or `exporter-storage-pod` if the `global.storagePod` parameter is enabled. **Note**: Ensure that the path provided here already exists on the Kubernetes slave name specified in `global.slaveNodename`.	/tmp	-
`global.capacityStorage`	Specifies the estimated amount of space to be occupied by the collected data, for example, 2Gi, 200Mi, and so on.	5Gi	1Gi, 4Gi, 500Mi, 10Gi, and so on
`global.storagePod`	Enables the storage pod mounted with persistence volume. When the value is set to `true`, a storage pod is placed and path provided in `global.outputPath` is mounted inside the pod at `/volume`. This pod can be used to copy the collected data without Kubernetes slave login.	false	true/ false
`global.elasticSearchURL`	Specifies the URL for Elastic search. FQDN of Elastic search can also be provided. If Elastic search requires authentication, it can be provided in the URL as: `http://<user-name>:<password>@<elastic-search-url>:<elastic-search-port>` **Note**: In case of OCCNE, use the elasticsearch-client FQDN.	-	-
`global.prometheusURL`	Specifies the URL for the Prometheus server. FQDN of Elastic search can also be provided. If the Prometheus server requires authentication, it can be provided in the URL as: `http://<user-name>:<password>@<prometheus-server-url>:<prometheus-server-port>`	-	-

Table 5-2 (Cont.) exporter-custom-values.yaml Parameters

Parameter	Description	Default Value	Range or Possible Value

`LogsExporter.inclusionFilters`	Provides comma separated json key values that must be present in the logs to be collected.
	For information about logs inclusion filter keywords, refer to NF Specific Logs Inclusion Filter Keywords.
	Example:
	`LogsExporter.inclusionFilters: |`
	`- vendor=oracle,application=ocnssf`
	`- vendor=oracle,application=ocnrf`
		-	-

`LogsExporter.exclusionFilters`	Specifies the list of field key values that must not exist in the logs to be generated. Example: `LogsExporter.exclusionFilters: | - audit=true`	-	-
`global.interval`	Specifies the interval in hours for which the data is required. The collected data will be last interval hours from now. Example: If the interval is set to 1, the data collector collects data for last one hour.	1	-
`global.pastTimeSpot [Optional Parameter]`	This parameter along with interval parameter specifies the time range for which the data has to be collected. This parameter accepts time in the UTC format. By default, the Exporter utility collects the last one hour data. This parameter can be used to override the default behavior. Example: If you want to generate the data from `2020-05-17T15:30:38Z" to "2020-05-17T17:30:38Z`, then you must set pastTimeSpot to `2020-05-17T17:30:38Z` and interval to 2. The system considers the current time to be `2020-05-17T17:30:38Z` and goes back 2 hours in time to collect the data ranging from `2020-05-17T15:30:38Z`to `2020-05-17T17:30:38Z`.	-	-
`MetricsExporter.step`	Provides the step in seconds to generate metrics data. When the step value is set to 30, it generates metrics data of each metric in an interval of 30 seconds.	30	-

Table 5-2 (Cont.) exporter-custom-values.yaml Parameters

Parameter	Description	Default Value	Range or Possible Value

`MetricsExporter.inclusionFilters`	Provides comma separated labels which must be present in the metrics to be collected.
	For information about metrics inclusion filter keywords, refer to NF Specific Metrics Inclusion Filter Keywords.
	Example:
	`TracesExporter.inclusionFilters: |`
	`- vendor=oracle,application=ocnssf`
	`- vendor=oracle,application=ocnrf`
		-	-

`TracesExporter.inclusionFilters`	Provides comma separated tags which must be present in the traces to be collected.
	Example:
	`TracesExporter.inclusionFilters: |`
	`- vendor=oracle,application=ocnssf`
	`- vendor=oracle,application=ocnrf`
		-	-

`TracesExporter.exclusionFilters`	Provides comma separated tags which must not be present in the traces to be collected.
	Example:
	`TracesExporter.exclusionFilters: |`
	`- audit=true`
		-	-

Sample custom file

```
global:
  # Registry where cnc-nfdata-collector image present.
  image:
  repository: reg-1:5000
  #Host machine is the slave node on which this job is scheduled. Make sure
path is present on node already.
  outputPath: /tmp
  # Name of the slave where fetched data can be kept.
  slaveNodeName: k8s-slave-node-1
  #Storage to be allocated to persistence
  capacityStorage: 30Gi
  #Mount slave path with pod
  storagePod: false
  #Mention the URL of elasticSearch here.#Mention the URL of elasticSearch
here.
  elasticSearchURL: "http://10.75.226.21:9200"
  #Mention the URL of prometheus here.
  prometheusURL: "http://10.75.226.49"
```

```
  #Time range for which data should fetched
  interval: "24" # IN HOURS
  #In case, data other than last few hours from now is required.
  #pastTimeSpot: "2020-05-17T15:30:38Z"
  LogsExporter:
  # Enable to fetch logs Data
  enabled: true
  # provide the list of json key values which must exist in the logs to be
fetched
  inclusionFilters: |
  - vendor=oracle,application=ocnssf
  # provide the list of json key values which must not exist in the logs to
be fetched
  exclusionFilters: |
  - audit_logs=true
  #Default REGEX value for this param is '^.*$' which means select all the
indices.
  match: '^.*$'
  MetricsExporter:
  # Enable to fetch Metrics Data
  enabled: true
  # provide the list of labels which must exist in the metrics to be fetched
  inclusionFilters: |
  - application=ocnssf
  # Timestamp difference between two data points in seconds
  step: "30"
  TracesExporter:
  # Enable to fetch Traces Data
  enabled: true
  # provide the list of tags which must exist in the traces to be fetched
  inclusionFilters: |
  - vendor=oracle,application=ocnssf
  # provide the list of labels which must not exist in the traces to be
fetched
   exclusionFilters: |
  - exclude_field=true
  #Default REGEX value for this param is '^.*$' which means select all the
indices.
  match: '^jaeger.*$'
```

5.2.3 Exporting the NF Logs, Metrics, Traces, and Alerts Data

Create the YAML file as described in Creating or Updating the YAML File for the Exporter Utility.

1. Run the following kubernetes job to collect the data:
   - For Helm 2:

     ```
     helm install ocnf-data-exporter/ --name <helm-release> --namespace <k8s
     namespace> -f <exporter_customized_values.yaml>
     ```

Example:

```
helm install ocnf-data-exporter/ --name ocnf-data-exporter --namespace
ocnf-data-exporter -f exporter-custom-values.yaml
```

- For Helm 3:

```
helm install <helm-release> ocnf-data-exporter/ --namespace <k8s
namespace> -f <exporter_customized_values.yaml>
```

Example:

```
helm install <helm-release> ocnf-data-exporter/ --namespace ocnf-data-
exporter -f exporter-custom-values.yaml
```

Where,

  - `<helm-release>` indicates the name provided by the user to identify the helm deployment.
  - `<k8s namespace>` indicates the name provided by the user to identify the kubernetes namespace of the utility. The job runs in this kubernetes namespace.

The file that is created or updated is used in the helm file.

2. Run the following command to check the status of the job:

```
helm status <helm-release>
```

Example:

```
helm status ocnf-data-exporter
```

3. Run the following command to check the status of the pods:

```
kubectl get pods -n <k8s namespace>
```

Ensure that the Status column of all the pods must be `Running` and the Ready column of all the pods must be `n/n`, where `n` indicates the number of containers in the pod.

Example:

```
kubectl get pods -n ocnf-data-exporter
NAME                                      READY     STATUS     RESTARTS
AGE
ocnf-data-exporter-dwdhwjw64vd3            3/3      Running      0
3m2s
```

4. Run the following command to check the completion of the job:

```
kubectl get pods -n <k8s namespace>
```

Ensure that the Status column of all the pods must be `Complete` and the Ready column of all the pods must be `0/n`, where `n` indicates the number of containers in the pod.

Example:

```
kubectl get pods -n ocnf-data-exporter
NAME                                        READY     STATUS      RESTARTS
AGE
ocnf-data-exporter-dwdhwjw64vd3             0/3       Complete      0
3m2s
```

5. If you want to remove the job after it is complete, then run the following commands:
   - For Helm 2:

     ```
     helm del --purge <helm-release>
     ```

     Example:

     ```
     helm del --purge ocnf-data-exporter
     ```

   - For Helm 3:

     ```
     helm uninstall <helm-release> -n <k8s namespace>
     ```

     Example:

     ```
     helm uninstall ocnf-data-exporter-n ocnf-data-exporter
     ```

Result

The Exporter utility creates the `exported-data_<current-date>_<current-time>` directory in the specified path in the directory provided in the `exporter-custom-values-<release number>.yaml` file parameter `global.outputPath`.

5.2.4 Sample Content of the Exporter Utility Custom values.yaml

Sample output:

```
[root@k8s-cluster-master tmp]# du -sh exported-data_2020-07-03_08:15:38/*
232K    exported-data_2020-07-03_08:15:38/logs
2.2G    exported-data_2020-07-03_08:15:38/metrics
88K     exported-data_2020-07-03_08:15:38/traces

[root@k8s-cluster-master tmp]# ls exported-data_2020-07-03_08:15:38/logs
jaeger-service-2020-06-28.json           jaeger-
service-2020-07-02.settings.json  jaeger-span-2020-07-02.json
logstash-2020.06.28.template.json  logstash-2020.07.01.settings.json
logstash-2042.01.06.json
jaeger-service-2020-06-28.mapping.json   jaeger-
service-2020-07-02.template.json  jaeger-span-2020-07-02.mapping.json
logstash-2020.06.29.mapping.json   logstash-2020.07.01.template.json
logstash-2042.01.06.mapping.json

[root@k8s-cluster-master tmp]# ls exported-data_2020-07-03_08:15:38/metrics
metrics-dump_ThreadPoolExecutor-0_0.json   metrics-
```

```
dump_ThreadPoolExecutor-0_16.json  metrics-dump_ThreadPoolExecutor-0_22.json
metrics-dump_ThreadPoolExecutor-0_29.json

[root@k8s-cluster-master tmp]# ls exported-data_2020-07-03_08:15:38/traces
jaeger-service-2020-06-28.json           jaeger-
service-2020-06-30.json          jaeger-service-2020-07-02.json
jaeger-span-2020-06-28.json          jaeger-span-2020-06-30.json
```

5.2.5 Transferring the Data Collected by Exporter Utility

Perform this procedure to transfer the collected data from the site on which the Exporter utility is present. If the storage pod and path provided in global.outputPath is mounted inside the pod at `/volume`, then this pod can be used to copy the collected data. The data must be extracted and move to a location where it can be reviewed.

1. Copy the collected data from the storage pod by running the following command:

   ```
   kubectl cp ocnf-data-exporter/exporter-storage-pod:/volume/exported-data ./
   exported-data
   ```

2. Create the tar of the collected data directory by running the following command:

   ```
   tar -cvzf exported-data.tgz exported-data/
   ```

3. After creating a tar, transfer the tar to the loader environment by running the following command:

   ```
   scp exported-data.tgz <user>@<machine-ip>:<path>
   ```

6

Viewing NF Deployment Data Collector Archive

- Unpack the archive provided by this module to view the information collected.

6.1 Loading the NF Logs, Metrics, Traces, Alerts Data

6.1.1 Creating or Updating the YAML File for the Loader Utility

The Loader utility YAML file is present in the CNC NF Data Collector tar package. You can either update the existing YAML file or create a new YAML file with the following parameters which can be configured. Oracle recommends to consider the existing file that is available with the tar package and update its parameters.
For information about parameter descriptions, refer to loader-custom-values.yaml Parameter Description.

If users of the Exporter utility wants to load the collected data to third-party tools to view them, then users can use the loader utility to load that data to third-party tools.

1. Update the `global.image.repository` parameter by providing the name of the docker registry where the `cnc-nfdata-collector` image is present.
2. Update the `global.slaveNodeName` parameter by providing the name of the kubernetes worker node that can load the collected data from this utility.

   To obtain the name of the worker node or slave node, run the `kubectl get nodes` command. The names of all the master and worker nodes of the kubernetes cluster are displayed in the Name column. You can provide the name of one of the worker nodes from the generated output.
3. Update the `global.inputPath` parameter by providing the path of the `exported-data` directory present on the kubernetes cluster worker node in `global.slaveNodeName`.

   - Ensure that the exported data is accessible by non-root users for read or write operation.
4. Update the `global.capacityStorage` parameter by specifying the estimated amount of space required to occupy the collected data, for example, 2Gi, 200Mi, and so on.

   The value specified here is the space provided to kubernetes persistence volume that is mounted with this utility to enable collected data accessible to loader job.
5. Update the `global.elasticSearchURL` parameter by specifying the URL for Elastic search.

   FQDN of Elastic search can also be provided. If Elastic search requires authentication, it can be provided in the URL as `http://<user-name>:<password>@<elastic-search-url>:<elastic-search-port>`.
   To find the FQDN, run `kubectl get svc –n <namespace>` and retrieve the Elastic search service name. After that, FQDN can be constructed as `http://<elastic-search-service-name>.<namespace>:<port>`. The default Elastic search port is 9200. The

administrator of the cluster must be asked for the user name and password of Elastic search if required.

6. Update the `global.victoriaMetricsURL` parameter by specifying the URL for the Victoria Metrics.

   FQDN of Victoria Metrics can also be provided.

6.1.1.1 loader-custom-values.yaml Parameter Description

The following table describes the parameters which can be customized while updating the `loader-custom-values.yaml` file.

Table 6-1 loader-custom-values.yaml Parameters

Parameter	Description	Default Value	Range or Possible Value
`global.image.repository`	Specifies the name of the docker registry that contains the cnc-nfdata-collector image.	-	-
`global.inputPath`	Specifies the name of the Kubernetes slave that contains the `exported-data/` directory. **Note**: The path must include `exported-data/`.	/tmp	-
`global.slaveNodeName`	Specifies the name of the Kubernetes slave that stores the collected data, for example, the `exported-data/` directory. To obtain the name of the slave node or worker node, run the `kubectl get nodes` command. Then, provide the name of one of the worker nodes as mentioned in the Name column of the generated output.	-	-
`global.capacityStorage`	Specifies the amount of space that is utilized by the collected data, for example, 2Gi, 200Mi, and so on.	5Gi	-
`global.elasticSearchURL`	Provides the URL for Elastic search. FQDN of Elastic search can also be provided. If Elastic search requires authentication, it can be provided in the URL as: `http://<user-name>:<password>@<elastic-search-url>:<elastic-search-port>`	-	-
`global.victoriaMetricsURL`	Provides the URL for Victoria metrics. FQDN of Victoria metrics can also be provided.	-	-

Sample custom file

```
global:
  # Registry where cnc-nfdata-collector image present.
 image:
    repository: reg-1:5000

  # path on slave node where exported-data folder is present. path including
the exported-data directory
 inputPath: /tmp/exported-data_2020-07-03_08:15:38

  #Mention the URL of elasticSearch here.
 elasticSearchURL: "https://
elastic:r2qrkb46sx6hh5zk56wcrshr@10.178.246.56:30785"

 #Mention the URL of victoria here.
 victoriaMetricsURL: "http://10.178.246.56:32001"

 #Storage to be allocated to persistence
 capacityStorage: 5Gi

  #Name of the slave where fetched data has to be loaded
 slaveNodeName: k8s-slave-node-1
```

6.1.2 Loading the NF Logs, Metrics, Traces, Alerts Data

Create the YAML file as described in Creating or Updating the YAML File for the Loader Utility.

1. Run the following commands to load the data:
   - For Helm 2:

     ```
     helm install ocnf-data-loader/ --name <helm-release> --namespace <k8s
     namespace> -f <loader_customized_values.yaml>
     ```

     Example:

     ```
     helm install ocnf-data-loader --name ocnf-data-loader --namespace ocnf-
     data-loader -f loader-custom-values.yaml
     ```

   - For Helm 3:

     ```
     helm install <helm-release> ocnf-data-loader/ --namespace <k8s
     namespace> -f <loader_customized_values.yaml>
     ```

     Example:

     ```
     helm install ocnf-data-loader ocnf-data-loader/ --namespace ocnf-data-
     loader -f loader-custom-values.yaml
     ```

     Where,

- <helm-release> indicates the name provided by the user to identify the helm deployment.
- <k8s namespace> indicates the name provided by the user to identify the kubernetes namespace of the utility. The job starts in this kubernetes namespace.

2. Run the following command to check the status of the job:

```
helm status <helm-release>
```

Example:

```
helm status ocnf-data-loader
```

3. Run the following command to check the status of the pods:

```
kubectl get pods -n <k8s namespace>
```

Ensure that the Status column of all the pods must be `Running` and the Ready column of all the pods must be `n/n`, where `n` indicates the number of containers in the pod.

Example:

```
kubectl get pods -n ocnf-data-loader
NAME                                    READY    STATUS    RESTARTS
AGE
ocnf-data-loader-dsgsf643tr              3/3     Running     0
1m2s
```

4. Run the following command to check the completion of the job:

```
kubectl get pods -n <k8s namespace>
```

Ensure that the Status column of all the pods must be `Complete` and the Ready column of all the pods must be `0/n`, where `n` indicates the number of containers in the pod.

Example:

```
kubectl get pods -n ocnf-data-loader
NAME                                    READY    STATUS     RESTARTS
AGE
ocnf-data-loader-dsgsf643tr              0/3     Complete     0
3m2s
```

5. If you want to remove the job after it is complete, then run the following commands:
   - For Helm 2:

     ```
     helm del --purge <helm-release>
     ```

     Example:

     ```
     helm del --purge ocnf-data-loader
     ```

- For Helm 3:

  ```
  helm uninstall <helm-release> -n <k8s namespace>
  ```

  Example:

  ```
  helm uninstall ocnf-data-loader -n ocnf-data-loader
  ```

6.1.3 Viewing the NF Logs, Metrics, and Traces

Perform the following procedure to view NF Logs, Metrics, and Traces.

1. To view logs, do the following:
   a. Open the Kibana GUI that is integrated with Elastic search by using the Kibana URL.
   b. Provide the index pattern to scan the indices.
   c. Navigate to the Discover section to view the logs.
2. Import the NF provided dashboards on Grafana to view metrics, otherwise follow these steps:
   a. Open the Grafana GUI that is integrated with Victoria Metrics by using the Grafana URL.
   b. Navigate to the Explore section and provide Prometheus as data source.
   c. Search for the required metrics in the **Search** box.
3. To view traces, do the following:
   a. Open the Jaeger GUI that is integrated with Elastic search by using the Jaeger URL.
   b. Select the service name and operation that you want to view.

> **Note**
>
> By default, Jaeger scans only the data of the past 3 days from Elastic search. If the data is older than 3 days, provide the `--es.max-span-age` parameter at the start of Jaeger. For example, `--es.max-span-age 720h0m0s` scans the data of the past 30 days.

6.1.4 Deleting the NF Logs, Traces, and Metrics

1. Run the following command to delete logs and traces from internal Elastic Search:

   ```
   curl -X DELETE -k "http://<elastic-search-host>:<elastic-searchport>/_all"
   -H "Content-Type: application/json"
   ```

   Example:

   ```
   curl -X DELETE -k "https://
   elastic:76qt2b7rz87ms2cs6fmb2c4l@10.178.246.56:30731/_all" -H "Content-
   Type: application/json"
   ```

2. Run the following command to delete metrics from internal Victoria Metrics:

   ```
   curl 'http://<victoria-metrics-URL>:<port>/api/v1/admin/tsdb/delete_series?
   match\[\]=<mertic-name>'
   ```

   Example:

   ```
   curl 'http://10.178.246.56:32001/api/v1/admin/tsdb/delete_series?match\
   [\]=http_requests_total'
   ```

   > **Note**
   >
   > To delete all the Victoria Metrics at once, run the `kubectl delete pod <pod-name> -n <namespace>` command.

Appendices

This section provides information about logs, metrics, and traces inclusion filter keywords.

A
NF Specific Logs Inclusion Filter Keywords

The following tables describe NRF and SCP log inclusion keywords.

Table A-1 NRF Logs Inclusion Filter Keywords

Attribute	Details	Sample Value	Possible Values (if applicable)
application	NF Application Name	ocnrf	-
microservice	Microservice name	ocnrf-nfdiscovery	<helm release name>-nfdiscovery, <helm release name>-nfregistration, <helm release name>-nfaccesstoken, <helm release name>-nfsubscription, <helm release name>-nrfauditor, <helm release name>-nrfconfiguration
vendor	Vendor of Product	oracle	-
namespace	Namespace of NRF	ocnrf	As per deployment

Table A-2 SCP Logs Inclusion Filter Keywords

Attribute	Details	Sample Value	Possible Values (if applicable)
kubernetes_namespace_name	Namespace of SCP	ocscp	As per deployment
vendor	Vendor of Product	oracle	-
kubernetes_pod_name	Name of the pod	scp	<name of the pod>

B

NF Specific Metrics Inclusion Filter Keywords

The following tables describe NRF and SCP metrics inclusion keywords.

Table B-1 NRF Metric Inclusion Filter Keywords

Attribute	Details	Sample Value	Possible Values (if applicable)
`application`	NF Application Name	`ocnrf`	-
`vendor`	Vendor of Product	`Oracle`	-

Table B-2 SCP Metric Inclusion Filter Keywords

Attribute	Details	Sample Value	Possible Values (if applicable)
`kubernetes_namespace`	Namespace of SCP	`ocscp`	As per deployment
`vendor`	Vendor of Product	`oracle`	-
`kubernetes_pod_name`	Name of the pod	`scp`	`<name of the pod>`

C

NF Specific Traces Inclusion Filter Keywords

The following table describes SCP trace inclusion keywords.

Table C-1 SCP Trace Inclusion Filter Keywords

Attribute	Details	Sample Value	Possible Values (if applicable)
`serviceName`	Name of service	`ocscp`	-