Oracle® Communications

Cloud Native Core Network Function Data Collector User's Guide

Release 1.0.0
F33206-02
August 2020

Oracle Communications Cloud Native Core Network Function Data Collector User's Guide, Release 1.0.0

F33206-02

Contents

My Oracle Support

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

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

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

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

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

What's New in This Guide

This is the first issue of this guide.

List of Figures

List of Tables

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.
- Execute 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
FQDN	Fully Qualified Domain Name
Gi	Gigabytes
K8s	Kubernetes
Mi	Megabytes
NF	Network Function
NRF	Network Repository Function
OCNRF	Oracle Communications Network Repository Function
OSDC	Oracle Software Delivery Cloud
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

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 deployment data:

- Helm deployment status
- Helm configuration used for deploying the NF
- Status of all the K8 resources used in the helm deployment
- Deployment description of all the K8 resources
- Kubernetes events

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

Execute this utility on the same system from which the data needs to be collected in plain files. 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

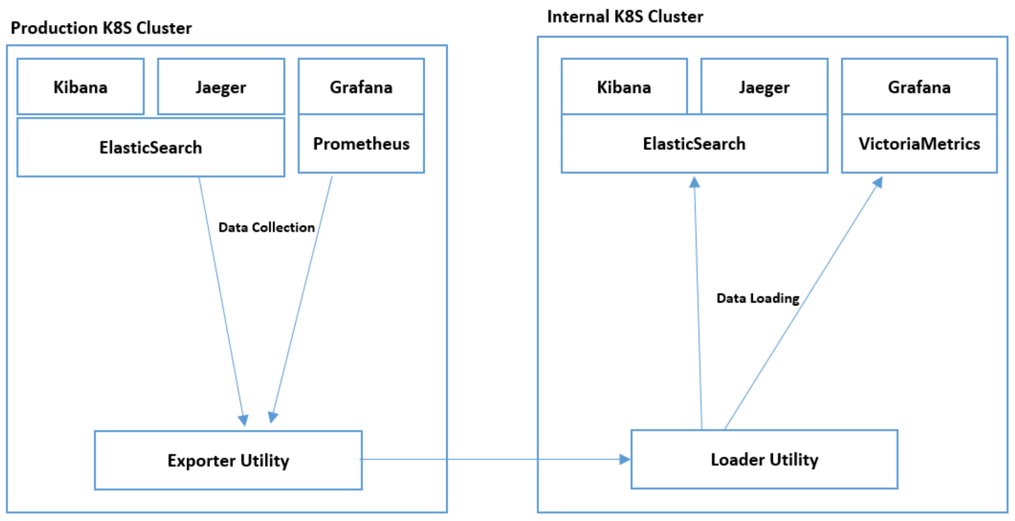
Loader Utility

Execute 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.

Prerequisites

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

- Kubernetes and helm must be installed on the host machine.
- 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.

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 for execution.

Ensure that you have completed the prerequisites as described in Prerequisites.

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

     ```
     ocnfDataCollector-pkg-1.0.0.0.0.tgz
     ```

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

     ```
     ocnfDataCollector-pkg-1.0.0.0.0-README.txt
     ```

2. Execute 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-1.0.0.0.0.tgz
              |_ _ _ _ _ _ _ ocnf-data-exporter-1.0.0.tgz (Helm Charts)
              |_ _ _ _ _ _ _ ocnf-data-loader-1.0.0.tgz (Helm Charts)
              |_ _ _ _ _ _ _ ocnfDataCollector-image-1.0.0.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.
4. Execute the following command to load the images to the local registry:

   ```
   docker load --input ocnfDataCollector-image-1.0.0.tar
   ```

5. Execute the `docker images` command to check whether all the images are loaded.
6. Execute 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>
   ```

7. Execute the following commands to untar the helm files:
   - `tar -xvzf <file name>`: To untar the `ocnf-data-exporter-1.0.0.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-1.0.0.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-1.0.0.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-1.0.0.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.

Executing the NF Deployment Data Collector Module

Perform this procedure to start the NF Deployment Data Collector module and generate the tarballs. This module generates the output in the same directory where the module is executed.

- Ensure that you must have appropriate privileges to access the system and execute 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 where the NF is deployed using helm.

1. Untar `ocnf-data-exporter-1.0.0.tgz.`

   The NF Deployment Data Collector module is extracted to the `ocnf-data-exporter` directory at `ocnf-data-exporter/nf-tools/dataCollector`. This directory also contains `ReadMe.txt`.
2. Execute the following command to start the module:

   ```
   ./nfDataCapture.sh -r|--release=[HELM_RELEASE_NAME] -k|--
   kubectl=[KUBE_SCRIPT_NAME] -h|--helm=[HELM_SCRIPT_NAME]  -s|--
   size=[SIZE_OF_EACH_TARBALL]
   ```

   Where,

   - `<HELM_RELEASE_NAME>` indicates the helm release number.
   - `<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.

   This module generates tarballs that starts with `<HELM_RELEASE_NAME>debugData`, which contains all the logs required for debugging, and then divides the same tar into several tarballs based on the size specified in the command. For more information, refer to Example 1 and Example 2. By default the size of each tarball is set to 10 megabytes if no size is specified in the command. For more information, refer to Example 3.
   Examples of the command:

- **Example 1**:

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

- **Example 2**:

  ```
  ./nfDataCapture.sh -r=ocnrf -s=5M
  ```

- **Example 3**:

  ```
  ./nfDataCapture.sh -r=ocnrf
  ```

Sample output

```
Data collection work completed. Packing data collected ....
Data collected to:- ./ocnrf.debugData.2020.07.02_09.27.19.tar.gz
```

3. After transferring the tarballs to the required destination, combine the tarballs into a single tarball by executing 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 ./ocnrf.debugData.2020.07.02_09.27.19-part* > ./
   ocnrf.debugData.2020.07.02_09.27.19-combined.tar.gz
   ```

Executing the NF Logs, Metrics, Traces, Alerts Data Collector Module

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 its 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, execute 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`. 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, execute the following kubectl cp command:

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

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, execute `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.

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.

   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.

    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.

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 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 nssf and 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
```

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 5-1 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.	-	-
`global.slaveNodeName`	Specifies the name of the k8s slave that stores the collected data. To obtain the name of the slave node or worker node, execute 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.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 k8s 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 k8s 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>`	-	-

Table 5-1 (Cont.) exporter-custom-values.yaml parameters

Parameter	Description	Default Value	Range or Possible Value
`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>`	-	-

`LogsExporter.inclusionFilters`	Provides comma separated json key values that must be present in the logs to be collected.
	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`	-	-
`LogsExporter.timeFilterFieldName`	Specifies the name of the time json field name.	@timestamp	-
`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	-

Table 5-1 (Cont.) exporter-custom-values.yaml parameters

Parameter	Description	Default Value	Range or Possible Value
`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 this 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`.	-	-
`LogsExporter.match`	Scans limited number of Elastic search indices by using regex. Use the default regex that can scan all the indices if the user is not aware of indices.	'^.*$'	-
`LogsExporter.limit`	Sets the number of documents that must be generated per request. This parameter can be used to enhance the performance. The value is limited to available buffer.	1000	-
`LogsExporter.nodeTLSRejectedUnauthorized`	Enables or disables certificate verification.	true	true/false
`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 5-1 (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.
	Example:
	`TracesExporter.inclusionFilters: |`
	`-`
	`vendor=oracle,application=ocnssf`
	`-`
	`vendor=oracle,application=ocnrf`
		-	-

`MetricsExporter.workerThreads`	Provides the number of worker threads to generate metrics data.	-	-
`TracesExporter.match`	Scans limited number of Elastic search indices by using regex. Jaeger traces are stored in index starting with jaeger by default. Use the default regex that can scan all the indices if the user is not aware of indices.	'^jaeger.*$'	-

`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`
		-	-

Table 5-1 (Cont.) exporter-custom-values.yaml parameters

Parameter	Description	Default Value	Range or Possible Value
`TracesExporter.unitOfTime`	Provides the unit of time to be used by time filter. This parameter provides epoch timestamp units. A 10 digit epoch has its units in seconds. A 13 digit epoch has its units in milliseconds. A 16 digit epoch has its units in microseconds. A 19 digit epoch has its units in nanosesonds. To know this value, you can count number of digits of the startTime field name.	microseconds	seconds, milliseconds, microseconds, and nanoseconds
`TracesExporter.timeFilterFieldName`	Provides the name of the time json field name.	startTime	-
`TracesExporter.limit`	Sets the number of documents that must be generated per request. This parameter can be used to enhance the performance. The value is limited to available buffer.	1000	-
`TracesExporter.nodeTLSRejectedUnauthorized`	Enables or disables certificate verification.	true	true/false

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: remote-setup-kamal
  #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
  # provide the name of the time json field name
  timeFilterFieldName: "@timestamp"
  #Default REGEX value for this param is '^.*$' which means select all
the indices.
  match: '^.*$'
  #Maximun number of records to be transferred in a batch
  limit: 1000
  #To disable certificate verification
  nodeTLSRejectedUnauthorized: "true"
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"
  # Number of worker threads to fetch metrics Data
  workerThreads: "32"
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
  # seconds, milliseconds, microseconds and nanoseconds.
  unitOfTime: "microseconds"
  # provide the name of the time json field name
  timeFilterFieldName: "startTime"
  #Default REGEX value for this param is '^.*$' which means select all
the indices.
  match: '^jaeger.*$'
  #Maximum number of records to be transferred in a batch
  limit: 1000
  #To disable certificate verification
  nodeTLSRejectedUnauthorized: "true"
```

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. Execute the following kubernetes job using helm to generate the data:

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

   ```
   helm install ocnf-data-exporter/ --name <helm-release> --namespace
   <k8s namespace> -f <exporter_customized_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 executes in this kubernetes namespace.

   Example:

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

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

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

   Example:

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

3. Execute 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. Execute 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
```

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`. 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
jaeger-service-2020-06-28.settings.json  jaeger-
span-2020-06-28.json              jaeger-span-2020-07-02.settings.json
logstash-2020.06.29.settings.json  logstash-2020.07.02.json
logstash-2042.01.06.settings.json
jaeger-service-2020-06-28.template.json  jaeger-
span-2020-06-28.mapping.json      jaeger-span-2020-07-02.template.json
logstash-2020.06.29.template.json  logstash-2020.07.02.mapping.json
logstash-2042.01.06.template.json
jaeger-service-2020-06-30.json           jaeger-
span-2020-06-28.settings.json     logstash-2020.06.27.json
logstash-2020.06.30.json            logstash-2020.07.02.settings.json
logstash-2062.01.03.json
jaeger-service-2020-06-30.mapping.json   jaeger-
span-2020-06-28.template.json     logstash-2020.06.27.mapping.json
logstash-2020.06.30.mapping.json   logstash-2020.07.02.template.json
logstash-2062.01.03.mapping.json
jaeger-service-2020-06-30.settings.json  jaeger-
span-2020-06-30.json              logstash-2020.06.27.settings.json
logstash-2020.06.30.settings.json  logstash-2020.07.03.json
```

```
logstash-2062.01.03.settings.json
jaeger-service-2020-06-30.template.json  jaeger-
span-2020-06-30.mapping.json      logstash-2020.06.27.template.json
logstash-2020.06.30.template.json  logstash-2020.07.03.mapping.json
logstash-2062.01.03.template.json
jaeger-service-2020-07-02.json           jaeger-
span-2020-06-30.settings.json     logstash-2020.06.28.mapping.json
logstash-2020.07.01.json           logstash-2020.07.03.settings.json
jaeger-service-2020-07-02.mapping.json   jaeger-
span-2020-06-30.template.json     logstash-2020.06.28.settings.json
logstash-2020.07.01.mapping.json   logstash-2020.07.03.template.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  metrics-
dump_ThreadPoolExecutor-0_6.json
metrics-dump_ThreadPoolExecutor-0_10.json
metrics-dump_ThreadPoolExecutor-0_17.json
metrics-dump_ThreadPoolExecutor-0_23.json
metrics-dump_ThreadPoolExecutor-0_2.json   metrics-
dump_ThreadPoolExecutor-0_7.json
metrics-dump_ThreadPoolExecutor-0_11.json
metrics-dump_ThreadPoolExecutor-0_18.json
metrics-dump_ThreadPoolExecutor-0_24.json
metrics-dump_ThreadPoolExecutor-0_30.json  metrics-
dump_ThreadPoolExecutor-0_8.json
metrics-dump_ThreadPoolExecutor-0_12.json
metrics-dump_ThreadPoolExecutor-0_19.json
metrics-dump_ThreadPoolExecutor-0_25.json
metrics-dump_ThreadPoolExecutor-0_31.json  metrics-
dump_ThreadPoolExecutor-0_9.json
metrics-dump_ThreadPoolExecutor-0_13.json
metrics-dump_ThreadPoolExecutor-0_1.json
metrics-dump_ThreadPoolExecutor-0_26.json  metrics-
dump_ThreadPoolExecutor-0_3.json
metrics-dump_ThreadPoolExecutor-0_14.json
metrics-dump_ThreadPoolExecutor-0_20.json
metrics-dump_ThreadPoolExecutor-0_27.json  metrics-
dump_ThreadPoolExecutor-0_4.json
metrics-dump_ThreadPoolExecutor-0_15.json
metrics-dump_ThreadPoolExecutor-0_21.json
metrics-dump_ThreadPoolExecutor-0_28.json  metrics-
dump_ThreadPoolExecutor-0_5.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           jaeger-span-2020-07-02.json
jaeger-service-2020-06-28.mapping.json
jaeger-service-2020-06-30.mapping.json   jaeger-
```

```
service-2020-07-02.mapping.json   jaeger-span-2020-06-28.mapping.json
jaeger-span-2020-06-30.mapping.json   jaeger-
span-2020-07-02.mapping.json
jaeger-service-2020-06-28.settings.json
jaeger-service-2020-06-30.settings.json  jaeger-
service-2020-07-02.settings.json  jaeger-span-2020-06-28.settings.json
jaeger-span-2020-06-30.settings.json  jaeger-
span-2020-07-02.settings.json
jaeger-service-2020-06-28.template.json
jaeger-service-2020-06-30.template.json  jaeger-
service-2020-07-02.template.json  jaeger-span-2020-06-28.template.json
jaeger-span-2020-06-30.template.json  jaeger-
span-2020-07-02.template.json
```

6

Viewing Collected Data

Viewing NF Deployment Data Collector Archive

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

Loading the NF Logs, Metrics, Traces, Alerts Data

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.

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, execute 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, execute `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.

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 k8s slave that contains the `exported-data/` directory. **Note**: The path must include `exported-data/`.	/tmp	-
`global.slaveNodeName`	Specifies the name of the k8s slave that stores the collected data, for example, the `exported-data/` directory. To obtain the name of the slave node or worker node, execute 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>`	-	-

Table 6-1 (Cont.) loader-custom-values.yaml parameters

Parameter	Description	Default Value	Range or Possible Value
`global.victoriaMetricsURL`	Provides the URL for Victoria metrics. FQDN of Victoria metrics can also be provided.	-	-
`global.limit`	Sets the number of documents that must be saved per request. This parameter can be used to enhance the performance. The value is limited to available buffer.	1000	-
`global.nodeTLSRejectedUnauthorized`	Enables or disables certificate verification.	true	true/false

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_2020-07-08_21:04:06
  #Mention the URL of elasticSearch here.
  elasticSearchURL: "https://
elastic:76qt2b7rz87ms2cs6fmb2c4l@10.178.246.56:30731"
  #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: remote-setup-kamal
  #To disable certificate verification
  nodeTLSRejectedUnauthorized: "true"
  #Maximum number of records to be transferred in a batch
  limit: "1000"
```

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. Execute the following kubernetes job to load the data:

   ```
   helm install ocnf-data-loader/ --name <helm-release> --namespace
   <k8s namespace> -f <loader_customized_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 executes in this kubernetes namespace.

Example:

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

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

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

   Example:

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

3. Execute 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. Execute 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
   ```

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. To view metrics, do the following:
   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.

Deleting Logs, Traces, and Metrics

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

   ```
   curl -XDELETE -k "http://<elastic-search-host>:<elastic-search-
   port>/_all" -H"Content-Type: application/json"
   ```

   Example:

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

2. Execute 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'
```