

Oracle® Communications

Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide



Release 26.1.201

G56076-01

August 2026

ORACLE®

G56076-01

Copyright © 2019, 2026, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

1 Introduction

1.1	Overview	1
1.2	References	2
1.3	Oracle Error Correction Policy	2
1.4	Oracle Open Source Support Policies	3

2 Installing BSF

2.1	Prerequisites	1
2.1.1	Software Requirements	1
2.1.2	Environment Setup Requirements	15
2.1.2.1	Client Machine Requirement	15
2.1.2.2	Network Access Requirement	16
2.1.2.3	Server or Space Requirement	17
2.1.2.4	CNE Requirement	17
2.1.2.5	cnDBTier Requirement	17
2.1.2.6	OSO Requirement	17
2.1.2.7	CNC Console Requirements	17
2.1.2.8	OCCM Requirements	17
2.1.3	Resource Requirements	18
2.1.3.1	Resource Requirements to Install BSF Services	18
2.1.3.2	Resource Requirements to Install BSF with ASM Sidecar	20
2.1.3.3	Resource Requirements for BSF Upgrade	21
2.2	Installation Sequence	22
2.2.1	Preinstallation Tasks	22
2.2.1.1	Verifying and Creating Namespace	22
2.2.1.2	Creating Service Account, Role, and RoleBinding	23
2.2.1.3	Configuring cnDBTier	25
2.2.1.4	Configuring Multiple Site Deployment	27
2.2.1.5	Creating Service Account, Role, and Role Binding for Helm Test	28
2.2.1.6	Configuring Database, Creating Users, and Granting Permissions	30
2.2.1.7	Configuring Kubernetes Secret for Accessing Database	44
2.2.1.8	Configuring Secrets for Enabling HTTPS	47
2.2.1.9	Configuring Secret for Enabling Access Token Validation	53

2.2.1.10	Configuring BSF to Support Aspen Service Mesh	60
2.2.1.11	Configuring Network Policies	69
2.2.1.12	Configuring Traffic Segregation	73
2.2.2	Installation Tasks	78
2.2.2.1	Downloading BSF Package	78
2.2.2.2	Pushing the Images to Customer Docker Registry	79
2.2.2.3	Installing BSF Package	82
2.2.3	Postinstallation Tasks	85
2.2.3.1	Verifying BSF Installation	85
2.2.3.2	Performing Helm Test	86
2.2.3.3	Backing Up Important Files	87
2.2.3.4	Configuring BSF Alerts	87

3 Customizing BSF

3.1	Configurations for Pre and Post Upgrade/Install Validations	1
3.2	Configuring Mandatory Parameters	5
3.3	Enabling or Disabling Services Configurations	6
3.4	Configuring Tracing Parameters	9
3.5	Configuring Database Names	14
3.6	Configuring Database Indexing	17
3.7	Configuring NRF client	19
3.8	Configuring Diameter Gateway	26
3.9	API Root Configuration for Notification URI	28
3.10	TLS Configurations	28
3.11	TLS Configuration in Diameter Gateway	33
3.12	Configuration for TLS Communication between Pod and Kubernetes API Server	35
3.13	Configuring Ingress Gateway	38
3.14	Configuring Egress Gateway	43
3.15	Configuring Service and Container Ports	51
3.16	OAuth Configuration	60
3.17	Configuring Ingress/Egress Gateway HTTPS	64
3.18	Configuring SCP	69
3.19	Logging Configuration	75
3.20	XFCC Header Validation Configuration	78
3.21	Aspen service mesh configurations	81
3.22	Alternate Route Service Configuration	84
3.23	Additional Configurations	86
3.24	Configurations for Metrics	89
3.25	Overload Manager Configurations	90
3.26	Configurable Error Codes	91
3.27	Server Header Configurations	93

3.28	Creating Custom Headers	94
3.29	Ingress Gateway Readiness Probe Configuration	95
3.30	Late Arrival Handling Configurations	97
3.31	Controlled Shutdown Configurations	100
3.32	Common Configurations for Services	102
3.33	Graceful Shutdown Configurations	112
3.34	Configurations for NodeSelector	114
3.35	Configuring Kafka for NF message feed	122
3.36	Configuration Parameters for Dual Stack and Single Stack IPv4 or IPv6	122

4 Upgrading BSF

4.1	Supported Upgrade Paths	2
4.2	Upgrade Strategy	2
4.3	Preupgrade Tasks	2
4.4	Upgrade Tasks	3
4.5	Postupgrade Tasks	7
4.5.1	Alert Configuration	7
4.6	MIB Management	7

5 Rolling Back BSF

5.1	Supported Rollback Paths	1
5.2	Rollback Tasks	1

6 Uninstalling BSF

6.1	Uninstalling CNC BSF using Helm	1
6.2	Deleting Kubernetes Namespace	1
6.3	Removing MySQL Users	2
6.4	Deleting PVC Volumes	2
6.5	Uninstalling Site in Georedundant Deployment	3
6.6	Uninstalling Last Site in Georedundant Deployment	4
6.6.1	Cleaning up NDB Replication Table	4
6.6.2	Cleaning up Databases	5

7 Fault Recovery

7.1	Overview	1
7.2	Impacted Areas	2
7.3	Prerequisites	3
7.4	Fault Recovery Scenarios	3

7.4.1	Scenario: Binding Database Corruption	4
7.4.1.1	When cnDBTier Failed in All Sites	4
7.4.2	Scenario: Site Failure	4
7.4.2.1	Single or Multiple Site Failure	5

A Resilient BSF Microservices with Kubernetes

B PodDisruptionBudget Configuration

C Docker Images

D Deployment Service Type Selection

Preface

- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Conventions](#)

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

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 in the following list on the Support telephone menu:

- 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**.
- For Hardware, Networking and Solaris Operating System Support, select **3**.

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 lists the acronyms and the terminologies used in the document:

Table Acronyms

Acronym	Description
3GPP	3rd Generation Partnership Project
AAA	Authorization Authentication Answer
AAR	Authorization Authentication Request
AF	Application Function
AMF	Access and Mobility Management Function
API	Application Programming Interface
ARS	Alternate Route Selection
ASM	Aspen Service Mesh
ASR	Abort-Session-Request
ATS	The core service sends the subscriber state variables to PDS only when there is an update to the variables.
AVP	Attribute Value Pair
BSF	Oracle Communications Cloud Native Core, Binding Support Function
CA	Certificate Authority
CDCS	Oracle Communications CD Control Server
CHF	Charging Function
CM	Configuration Management
CNC	Cloud Native Core
CNC Console	Oracle Communications Cloud Native Configuration Console
CNE	Oracle Communication Cloud Native Core, Cloud Native Environment
CNLB	Cloud Native Load Balancer
CNPCRf	Oracle Communications Cloud Native Core, Policy and Charging Rules Function
CUSTOMER_REPO	Docker registry address including the port number, if the docker registry has an associated port.
cnDBTier	Oracle Communications Cloud Native Core, cnDBTier
DNS	Domain Name System
DRA	Diameter Routing Agent
FQDN	Fully Qualified Domain Name
GUAMI	Globally Unique AMF Identifier
IMAGE_TAG	Image tag from release tar file. You can use any tag number. However, make sure that you use that specific tag number while pushing docker image to the docker registry.
IMS	IP Multimedia Subsystem
HTTPS	Hypertext Transfer Protocol Secure
LDAP	Lightweight Directory Access Protocol.
MCC	Mobile Country Code
MCPTT	Mission-critical push-to-talk
METALLB_ADDRESS_POOL	Address pool configured on metallb to provide external IPs

Table (Cont.) Acronyms

Acronym	Description
MNC	Mobile Network Code
NAD	Network Attachment Definitions
NEF	Oracle Communications Cloud Native Core, Network Exposure Function
NF	Network Function
NPLI	Network Provided Location Information
NRF	Oracle Communications Cloud Native Core, Network Repository Function
OSO	Oracle Communications Operations Services Overlay
P-CSCF	Proxy Call Session Control Function
PA Service	Policy Authorization Service
PCC	Policy and Charging Control
PDB	Pod Disruption Budget
PLMN	Public Land Mobile Network
PCF	Oracle Communications Cloud Native Core, Policy Control Function
PCRF	Oracle Communications Cloud Native Core, Policy and Charging Rules Function
PCEF	Policy and Charging Enforcement Function
PCSCF	Proxy Call Session Control Function
PDS	Policy Data Service
PRA	Presence Reporting Area
PRE	Policy Runtime Engine
PDU	Protocol Data Unit
Policy	Oracle Communications Cloud Native Core, Converged Policy
QoS	Quality of Service
RAA	Re-Auth-Answer
RAN	Radio Access Network
RAR	Re-Auth-Request
RedCap devices	Reduced Capability devices (Examples: Wearables, Wireless sensors, and Surveillance equipment)
SBI	Service Based Interface
SAN	Subject Alternate Name
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
SMF	Session Management Function
S-NSSAI	Single Network Slice Selection Assistance Information
UDR	Oracle Communications Cloud Native Core, Unified Data Repository
SRA	Successful Resource Allocation
STR	Session Termination Request
TTL	Time To Live
UE	User Equipment
UPF	User Plane Function
UPSI	UE Policy Section Identifier
URSP	UE Route Selection Policies
UPSC	UE Policy Section Code

Table (Cont.) Acronyms

Acronym	Description
URI	Uniform Resource Identifier
VoLTE	Voice over Long-Term Evolution
VoNR	Voice over New Radio
VSA	Vendor Specific Attributes

What's New in This Guide

This section introduces the documentation updates for release 26.1.2xx.

Release 26.1.201 - G56076-01, August 2026

Generic Updates:

- Updated the release version number to 26.1.201.
- Updated [Oracle Error Correction Policy](#) section for BSF 26.1.201.
- Updated the [Software Requirements](#) section with the third party software details for BSF 26.1.201.
- Updated the [CNE Requirement](#) section with the supported CNE versions for BSF 26.1.201.
- Updated the [cnDBTier Requirement](#) section with the supported cnDBTier version for BSF 26.1.201.
- Updated the [CNC Console Requirements](#) section with the supported CNC Console version for BSF 26.1.201.
- Updated the [OSO Requirement](#) section with the supported OSO version for BSF 26.1.201.

Installation Updates:

- Updated the [Pushing the Images to Customer Docker Registry](#) section to describe the compatible docker image versions for various BSF microservices.
- Added the parameter `bsfManagementService.metrics.altLookupTable.latency.percentiles`, which is used to configure the Percentiles of Latency metrics, to [Configuring Database Indexing](#) section under [Customizing BSF](#).

Upgrade and Rollback Updates:

- Updated the Supported Upgrade Path in [Supported Upgrade Paths](#).
- Updated the Supported Rollback Path in [Supported Rollback Paths](#).

1

Introduction

This guide describes how to install or upgrade Oracle Communications Cloud Native Core, Binding Support Function (BSF) in a cloud native environment.

Note

- This guide covers the installation instructions when Podman is the container platform with Helm as the Packaging Manager. For any other container platform, the operator must use the commands based on their deployed container runtime environment.
- `kubectl` commands can vary based on the platform deployment. Replace `kubectl` with Kubernetes environment-specific command line tool to configure Kubernetes resources through kube-api server. The instructions provided in this document are as per the CNE version of kube-api server.

Caution

User, computer and applications, and character encoding settings can cause an issue when copy-pasting commands or any content from PDF. PDF reader version also affects the copy-pasting functionality. It is recommended to verify the pasted content especially when the hyphens or any special characters are part of the copied content.

1.1 Overview

BSF is a key component of the 5G Service Based Architecture (SBA). BSF allows Policy Control Function (PCF) to register, update, and remove the binding information from it, and allows Network Function (NF) consumers to discover the selected Policy Control Function. BSF stores the binding information for certain Protocol Data Unit (PDU) sessions and discovers the selected PCF according to the binding information. It also acts as diameter proxy agent or diameter redirect agent to Rx requests targeting an IP address of a User Equipment (UE) to the selected Policy Control Function.

For any Application Function (AF) using Rx, such as Proxy-Call Session Control Function (P-CSCF), BSF determines the selected Policy Control Function address according to the information carried by the incoming Rx requests.

Note

The performance and capacity of the BSF system may vary based on the call model, Feature/Interface configuration, and underlying CNE and hardware environment.

BSF service allows:

- Policy Control Function users to register, update, and remove the binding information.
- NF consumers to retrieve the binding information.

For more information, see *Oracle Communications Cloud Native Core, Binding Support Function User Guide*.

1.2 References

Refer to the following documents while deploying BSF:

- *Oracle Communications Cloud Native Core, Binding Support Function User Guide*
- *Oracle Communications Cloud Native Core, Binding Support Function REST Specification Guide*
- *Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*
- *Oracle Communications Cloud Native Core, Binding Support Function Network Impact Report*
- *Oracle Communications Cloud Native Core, cnDBTier User Guide*
- *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Data Collector User Guide*
- *Oracle Communications Cloud Native Core Automated Test Suite Guide*
- *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*
- *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, OCI Deployment Guide*

1.3 Oracle Error Correction Policy

The table below outlines the key details for the current and past releases, their General Availability (GA) dates, and the end dates for the Error Correction Grace Period.

Table 1-1 Oracle Error Correction Policy

Release Number	General Availability (GA) Date	Error Correction Grace Period End Date
3.26.1.200.0	August 2026	August 2027
3.25.2.200.0	March 2026	March 2027
3.25.2.100.0	November 2025	November 2026
3.25.1.200.0	July 2025	July 2026

Note

- For the latest patch releases, see their corresponding *Oracle Communications Cloud Native Core Release Notes*.
- For a release, Sev1 and Critical Patch Update (CPU) patches are supported for 12 months. For more information, see [Oracle Communications Cloud Native Core and Network Analytics Error Correction Policy](#).

1.4 Oracle Open Source Support Policies

Oracle Communications Cloud Native Core uses open source technology governed by the Oracle Open Source Support Policies. For more information, see [Oracle Open Source Support Policies](#).

2

Installing BSF

This chapter provides information about installing Oracle Communications Cloud Native Core, Binding Support Function (BSF) in a cloud native environment.

Note

- BSF supports fresh installation, and it can also be upgraded from 25.2.2xx. For more information on how to upgrade BSF, see [Upgrading BSF](#).

2.1 Prerequisites

Before installing and configuring BSF, ensure that the following prerequisites are met.

2.1.1 Software Requirements

This section lists the software to be installed before installing BSF:

Note

[2-1](#) and [2-2](#) offer a comprehensive list of software necessary for the proper functioning of BSF during deployment. However, these tables are indicative, and the software used can vary based on the customer's specific requirements and solution.

The **Software Requirement** column in [2-1](#) and [2-2](#) indicates one of the following:

- **Mandatory:** Absolutely essential; the software cannot function without it.
- **Recommended:** Suggested for optimal performance or best practices but not strictly necessary.
- **Conditional:** Required only under specific conditions or configurations.
- **Optional:** Not essential; can be included based on specific use cases or preferences.

Table 2-1 Preinstalled Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Kubernetes	1.35.2	1.34.2	Mandatory	Kubernetes orchestrates scalable, automated NF deployments for high availability and efficient resource utilization. Note: BSF can be deployed in a Kubernetes cluster that supports TLS 1.3. Impact: Preinstallation is required. Without orchestration capabilities, deploying and managing network functions (NFs) can become complex, leading to inefficient resource utilization and potential downtime.
Helm	3.20.0	3.19.0	Mandatory	Helm, a package manager, simplifies deploying and managing NFs on Kubernetes with reusable, versioned charts for easy automation and scaling. Impact: Preinstallation is required. Without this capability, management of NF versions and configurations becomes time-consuming and error-prone, impacting deployment consistency.

Table 2-1 (Cont.) Preinstalled Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Podman	5.6.0	5.4.0	Recommended	Podman is a part of Oracle Linux. It manages and runs containerized NFs without requiring a daemon, offering flexibility and compatibility with Kubernetes. Impact: Preinstallation is required. Without efficient container management, the development and deployment of NFs could become cumbersome, impacting agility.

To check the Helm, Kubernetes, and Podman version installed, run the following commands:

```
helm version
```

```
kubectl version
```

```
podman version
```

① Note

This guide covers the installation instructions for BSF when Podman is the container platform with Helm as the Packaging Manager. For non-CNE, the operator can use commands based on their deployed Container Runtime Environment, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

If you are deploying BSF in a cloud native environment, these following additional software are to be installed before installing BSF.

Table 2-2 Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
AlertManager	0.30.0	0.28.0	Recommended	Alertmanager is a component that works in conjunction with Prometheus to manage and dispatch alerts. It handles the routing and notification of alerts to various receivers. Impact: Not implementing alerting mechanisms can lead to delayed responses to critical issues, potentially resulting in service outages or degraded performance.
Calico	3.30.6	3.30.3	Recommended	Calico provides networking and security for NFs in Kubernetes, ensuring scalable, policy-driven connectivity. Impact: Calico is a popular Container Network Interface (CNI) and CNI is mandatory for the functioning of 5G NFs. Without a CNI plugin, the network could witness security vulnerabilities and inadequate traffic management, impacting the reliability of NF communications.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
cinder-csi-plugin	1.35.0	1.33.0	Mandatory	Cinder CSI (Container Storage Interface) plugin is used for provisioning and managing block storage in Kubernetes. It is often used in OpenStack environments to provide persistent storage for containerized applications Impact: Without the CSI plugin, provisioning block storage for NFs would be manual and inefficient, complicating storage management.
containerd	2.2.1	2.1.4	Recommended	Containerd manages container lifecycles to run NFs efficiently in Kubernetes. Impact: A lack of a reliable container runtime could lead to performance issues and instability in NF operations.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
CoreDNS	1.12.4	1.12.0	Recommended	CoreDNS is the DNS server in Kubernetes, which provides DNS resolution services within the cluster. Impact: DNS is an essential part of deployment. Without proper service discovery, NFs would struggle to communicate with each other, leading to connectivity issues and operational failures.
Fluentd	1.19.0	1.17.1	Recommended	Fluentd is an open source data collector that streamlines data collection and consumption, ensuring improved data utilization and comprehension. Impact: Not utilizing centralized logging can hinder the ability to track NF activity and troubleshoot issues effectively, complicating maintenance and support.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Grafana	7.5.17-1.0.4	7.5.17(OCI Grafana)	Recommended	Grafana is a popular open source platform for monitoring and observability. It provides a user-friendly interface for creating and viewing dashboards based on various data sources. Impact: Without visualization tools, interpreting complex metrics and gaining insights into NF performance would be cumbersome, affecting effective management.
Jaeger	2.14.1	1.72.0	Recommended	Jaeger provides distributed tracing for 5G NFs, enabling performance monitoring and troubleshooting across microservices. Impact: Not utilizing distributed tracing may hinder the ability to diagnose performance bottlenecks, making it challenging to optimize NF interactions and user experience.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Kyverno	1.16.1	1.15.0	Recommended	Kyverno is a Kubernetes policy engine that allows to manage and enforce policies for resource configurations within a Kubernetes cluster. Impact: Without the policy enforcement, there could be misconfigurations, resulting in security risks and instability in NF operations, affecting reliability.
MetallB	0.15.2	0.15.2	Recommended	MetallB is used as a load balancing solution in CNE, which is mandatory for the solution to work. MetallB provides load balancing and external IP management for 5G NFs in Kubernetes environments. Impact: Without load balancing, traffic distribution among NFs may be inefficient, leading to potential bottlenecks and service degradation.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
metrics-server	0.8.0	0.7.2	Recommended	Metrics server is used in Kubernetes for collecting resource usage data from pods and nodes. Impact: Without resource metrics, auto-scaling and resource optimization would be limited, potentially leading to resource contention or underutilization.
Multus	4.2.2	4.2.1-thick	Recommended	Multus enables multiple network interfaces in Kubernetes pods, allowing custom configurations and isolated paths for advanced use cases such as NF deployments, ultimately supporting traffic segregation. Impact: Without this capability, connecting NFs to multiple networks could be limited, impacting network performance and isolation.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Oracle OpenSearch	2.19.1	2.18.0	Recommended	OpenSearch provides scalable search and analytics for 5G NFs, enabling efficient data exploration and visualization. Impact: Without a robust analytics solution, there would be difficulties in identifying performance issues and optimizing NF operations, affecting overall service quality.
OpenSearch Dashboard	2.19.1	2.18.0	Recommended	OpenSearch dashboard visualizes and analyzes data for 5G NFs, offering interactive insights and custom reporting. Impact: Without visualization capabilities, understanding NF performance metrics and trends would be difficult, limiting informed decision making.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Prometheus	3.9.0	3.6.0	Mandatory	Prometheus is a popular open source monitoring and alerting toolkit. It collects and stores metrics from various sources and allows for alerting and querying. Impact: Not employing this monitoring solution could result in a lack of visibility into NF performance, making it difficult to troubleshoot issues and optimize resource usage.
prometheus-kube-state-metric	2.18.0	2.16.0	Recommended	Kube-state-metrics is a service that generates metrics about the state of various resources in a Kubernetes cluster. It's commonly used for monitoring and alerting purposes. Impact: Without these metrics, monitoring the health and performance of NFs could be challenging, making it harder to proactively address issues.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
prometheus-node-exporter	1.10.2	1.9.1	Recommended	Prometheus Node Exporter collects hardware and OS-level metrics from Linux hosts. Impact: Without node-level metrics, visibility into infrastructure performance would be limited, complicating the identification of resource bottlenecks.
Prometheus Operator	0.88.1	0.83.0	Recommended	The Prometheus Operator is used for managing Prometheus monitoring systems in Kubernetes. Prometheus Operator simplifies the configuration and management of Prometheus instances. Impact: Not using this operator could complicate the setup and management of monitoring solutions, increasing the risk of missed performance insights.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
rook	1.17.7	1.16.7	Mandatory	Rook is the Ceph orchestrator for Kubernetes that provides storage solutions. It is used in BareMetal CNE solution. Impact: Not utilizing rook could increase the complexity of deploying and managing ceph, making it difficult to scale storage solutions in a Kubernetes environment.
snmp-notifier	2.0.0	2.0.0	Recommended	snmp-notifier sends SNMP alerts for 5G NFs, providing real-time notifications for network events. Impact: Without SNMP notifications, proactive monitoring of NF health and performance could be compromised, delaying response to critical issues.

Table 2-2 (Cont.) Additional Software Versions

Software	26.1.2xx	25.2.2xx	Software Requirement	Usage Description
Velero	1.17.1	1.13.2	Recommended	Velero backs up and restores Kubernetes clusters for 5G NFs, ensuring data protection and disaster recovery. Impact: Without backup and recovery capabilities, customers would witness a risk of data loss and extended downtime, requiring a full cluster reinstall in case of failure or upgrade.

To check the installed software, run the following command:

```
helm ls -A
```

! Important

If you are using NRF with BSF, install it before proceeding with the BSF installation. BSF 26.1.201 supports NRF 26.1.2xx.

Dashboard Modifications for Grafana 7.5.14

Change Area	Action Required	Examples
__inputs for DS_EXPRESSION	Remove	
__elements	Remove or set to {}	
__requires	Optional (can be removed)	
schemaVersion	Set to 27	"schemaVersion": 27
pluginVersion	Set to "7.5.17"	"pluginVersion": "7.5.17"
datasource	Replace UID-based config with null	Update - "datasource": { "type": "prometheus", "uid": "\${DS_PROMXY}" } To - "datasource": null

legendFormat	Replace "__auto" with label-based formatting	If metric expression is "expr": "sum by (pod)(any_metric)" Update - "legendFormat": "__auto" To- "legendFormat": "{{pod}}" Note: Update according to the metric expression (what needs to be visualised).
Panel Type (graph)	Switch to "timeseries" (recommended)	Modify visualization type to timeseries. Recommended: Change it from UI and use template json generated from grafana.
Template fixing	Move sections outside options{ }. (Majorly issue has been seen for override mappings)	If panel settings defined in the JSON may not appear in the UI, may be they are wrongly placed. Move those properties outside options{ }.

Note

- Each panel must have a unique ID; otherwise, panels may disappear after row collapsing.
- The 'last*' legend option in version 9.5.3 is equivalent to the 'last (not null)' option in 7.5.14; the difference is only in the user interface (UI).
- In version 7.5.14, to visualise data as a heatmap with buckets, you need to explicitly format it as time series buckets. This feature was rewritten in version 9.x, and the time series buckets option is no longer available.
- The 7.5.14 version does not support the tooltip option for pie charts, so the data cannot be represented in the same way as in the Grafana 9.5.3 version.

Note

Table header may not render when there is no data, due to an inconsistency in the rendering logic in 7.5.14 version. It was addressed in subsequent versions, where the header rendering logic was improved to ensure consistent display.

- The y-axis scale on a time series panel may not always begin at zero. To achieve this visualization, you can specifically set the soft minimum value to zero.

2.1.2 Environment Setup Requirements

This section describes the environment setup requirements required for installing BSF.

2.1.2.1 Client Machine Requirement

This section describes the requirements for client machine, that is, the machine used by the user to run deployment commands.

The client machine must have:

- Helm repository configured on the client.
- network access to the Helm repository and Docker image repository.
- network access to the Kubernetes cluster.

Note

BSF can be deployed in a Kubernetes cluster that supports TLS 1.3.

- required environment settings to run the `kubectl`, `podman`, and `docker` commands. The environment should have privileges to create namespace in the Kubernetes cluster.
- Helm client installed with the push plugin. The environment should be configured so that the `helm install` command deploys the software in the Kubernetes cluster.

2.1.2.2 Network Access Requirement

The Kubernetes cluster hosts must have network access to the following repositories:

- **Local Helm Repository:** It contains the BSF Helm charts.
To check if the Kubernetes cluster hosts have network access to the local Helm repository, run the following command:

```
helm repo update
```

- **Local Docker Image Repository:** It contains the BSF Docker images.
To check if the Kubernetes cluster hosts can access the local Docker image repository, pull any image with an `image-tag`, using the following command:

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

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

Where:

- `<docker-repo>` is the IP address or host name of the Docker repository.
- `<podman-repo>` is the IP address or host name of the Podman repository.
- `image-name` is the Docker image name.
- `image-tag` is the tag assigned to the Docker image used for the BSF pod.

For Example:

```
docker pull CUSTOMER_REPO/oc-app-info:26.1.205
```

For CNE 1.8.0 and later versions, use the following command:

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

For Example:

```
podman pull CUSTOMER_REPO/oc-app-info:26.1.205
```

Note

Run the `kubectl` and `Helm` commands on a system based on the deployment or infrastructure. For instance, you can run these commands on a client machine such as VM, server, local desktop, and so on.

2.1.2.3 Server or Space Requirement

For information about server or space requirements, see the *Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) Installation, Upgrade and Fault Recovery Guide*.

2.1.2.4 CNE Requirement

This section is applicable only if you are installing BSF on Cloud Native Environment (CNE). BSF supports CNE 26.1.2xx and 25.2.2xx.

To check the CNE version, run the following command:

```
echo $OCNE_VERSION
```

For more information, see *Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) Installation, Upgrade, and Fault Recovery Guide*.

2.1.2.5 cnDBTier Requirement

BSF supports cnDBTier 26.1.2xx and 25.2.2xx. cnDBTier must be configured and running before installing BSF. For more information about cnDBTier installation, see *Oracle Communications Cloud Native Core, cnDBTier (cnDBTier) Installation, Upgrade, and Fault Recovery Guide*.

2.1.2.6 OSO Requirement

BSF supports Operations Services Overlay (OSO) 26.1.2xx and 25.2.2xx for common operation services (Prometheus and components such as Alertmanager, Pushgateway) on a Kubernetes cluster, which does not have these common services. For more information on installation procedure, see *Oracle Communications Cloud Native Core, Operations Services Overlay Installation, Upgrade, and Fault Recovery Guide*.

2.1.2.7 CNC Console Requirements

BSF supports CNC Console (CNCC) 26.1.2xx and 25.2.2xx.

For more information about CNCC, see *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide* and *Oracle Communications Cloud Native Configuration Console User Guide*.

2.1.2.8 OCCM Requirements

To support automated certificate lifecycle management, BSF integrates with Oracle Communications Cloud Native Core, Certificate Management (OCCM) in compliance with 3GPP security recommendations. For more information about OCCM in BSF, see the *Support*

for *Automated Certificate Lifecycle Management* section in *Oracle Communications Cloud Native Core, Binding Support Function User Guide*.

For more information about OCCM, see the following guides:

- *Oracle Communications Cloud Native Core, Certificate Manager Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Certificate Manager User Guide*

2.1.3 Resource Requirements

This section lists:

- Resource requirements to install BSF services
- Resource requirements to install BSF services with ASM Sidecar
- Resource requirements to upgrade BSF

Note

The performance and capacity of the BSF system may vary based on the call model, Feature/Interface configuration, and underlying CNE and hardware environment.

2.1.3.1 Resource Requirements to Install BSF Services

The following table lists resource requirements to install BSF services:

Note

For resource requirements details on BSF installation with ASM Sidecar, see [Resource Requirements to Install BSF with ASM Sidecar](#).

Table 2-3 Resource requirements to install BSF services

Service	CPU Request per Pod (#)	CPU Limit per Pod (#)	Memory Request per Pod (GB)	Memory Limit per Pod (GB)	Min Replicas (#)	Max Replicas (#)	Ephemeral Storage Request per Pod (MB)	Ephemeral Storage Limit per Pod (GB)
bsf-management-service	4	4	8	8	2	2	80	4

To modify the default values of replicas for any BSF microservice, you must add the following parameters under the required service group with CPU and memory in `ocbsf_custom_values_26.1.201.yaml` file:

```
minReplicas: 1
maxReplicas: 1
```

For example, to update the default values for Ingress gateway or Egress gateway, add the parameters under ingress-gateway or egress-gateway group:

```
ingress-gateway:
  #Resource details
  resources:
    limits:
      cpu: 1
      memory: 6Gi
    requests:
      cpu: 1
      memory: 2Gi
    target:
      averageCpuUtil: 80
  minReplicas: 1
  maxReplicas: 1
egress-gateway:
  #Resource details
  resources:
    limits:
      cpu: 1
      memory: 6Gi
    requests:
      cpu: 1
      memory: 2Gi
    target:
      averageCpuUtil: 80
  minReplicas: 1
  maxReplicas: 1
```

Note

It is recommended to avoid altering the above mentioned standard resources. Either increasing or decreasing the CPU or memory will result in unpredictable behavior of the pods. Contact My Oracle Support (MOS) team for Min Replicas and Max Replicas count values.

The following table lists resource requirement for common services to deploy BSF:

Table 2-4 Common Services Container

Service	CPU Request per Pod (#)	CPU Limit per Pod (#)	Memory Request per Pod (GB)	Memory Limit per Pod (GB)	Min Replicas (#)	Max Replicas (#)	Ephemeral Storage Request per Pod (MB)	Ephemeral Storage Limit per Pod (GB)
AppInfo	2	2	1	1	2	2	80	1
Audit Service	2	2	1	1	2	2	80	1
CM Service	4	4	2	2	2	2	80	2
Config Server	4	4	2	2	2	2	80	2

Table 2-4 (Cont.) Common Services Container

Service	CPU Request per Pod (#)	CPU Limit per Pod (#)	Memory Request per Pod (GB)	Memory Limit per Pod (GB)	Min Replicas (#)	Max Replicas (#)	Ephemeral Storage Request per Pod (MB)	Ephemeral Storage Limit per Pod (GB)
Diameter Gateway	4	4	2	2	2	2	80	2
Egress Gateway	4	4	6	6	2	2	80	6
Ingress Gateway	4	4	6	6	2	2	80	6
NRF Client NF Discovery	4	4	2	2	2	5	80Mi	1Gi
NRF Client NF Management	1	1	1	1	2	2	80	1
Query Service	2	2	1	1	2	2	80	1
PerfInfo	4	4	1	1	2	2	80	1
Alternate Route Service	2	2	2	4	2	2	80	4

Note

Max replica per service should be set based on required TPS and other dimensioning factors.

You must take Upgrade resources into account during dimensioning. Default upgrade resource requirements are 25% above max replica, rounding up to the next integer. For example, if a service has a max replica count of 8, upgrade resources of 25% will result in additional resources equivalent to 2 pods. If a max replica is 1, one additional pod would be required (rounding 0.25 to 1).

2.1.3.2 Resource Requirements to Install BSF with ASM Sidecar

BSF leverages the Platform Service Mesh (for example, Aspen Service Mesh) for all internal and external TLS communication. If ASM Sidecar injection is enabled during BSF installation or upgrade, this container is injected to each BSF pod (or selected pod, depending on the option chosen during installation or upgrade). These containers stay till pod or deployment exist. For more information about installing ASM, see [Configuring BSF to Support Aspen Service Mesh](#).

Table 2-5 Resource requirements to install BSF with ASM Sidecar

Service	CPU Request per Pod (#)	CPU Limit per Pod (#)	Memory Request per Pod (GB)	Memory Limit per Pod (GB)	Min Replicas (#)	Max Replicas (#)	Ephemeral Storage Request per Pod (MB)	Ephemeral Storage Limit per Pod (GB)	Istio CPU Request per Pod (#)	Istio CPU Limit per Pod (#)	Istio Memory Request per Pod (Min)	Istio Memory Limit per Pod (Max)
bsf-management-service	4	4	8	8	2	2	80	4	2	2	2	2
Alternate Route Service	2	2	2	4	2	2	80	4	2	2	2	2
Diameter Gateway	4	4	2	2	2	2	80	2	2	2	2	2
Egress Gateway	4	4	6	6	2	2	80	6	2	2	2	2
Ingress Gateway	4	4	6	6	2	2	80	6	2	2	2	2
NRF Client NF Management	1	1	1	1	2	2	80	1	2	2	2	2
Perf-Info	4	4	1	1	2	2	80	1	2	2	2	2
App-info	2	2	1	1	2	2	80	1	2	2	2	2
Query Service	2	2	1	1	2	2	80	1	2	2	2	2
CM Service	4	4	2	2	2	2	80	2	2	2	2	2
Config Server	4	4	2	2	2	2	80	2	2	2	2	2
Audit Service	2	2	1	1	2	2	80	1	2	2	2	2

2.1.3.3 Resource Requirements for BSF Upgrade

Following is the resource requirement for upgrading BSF.

Table 2-6 Upgrade

Service	CPU		Memory (Gi)		Replica(s)
	Min	Max	Min	Max	Max
Alternate Route Service	1	2	1	2	1
bsf-management-service	1	2	1	2	1
Diameter Gateway	1	2	1	2	1
Egress Gateway	1	2	1	2	1
Ingress Gateway	1	2	1	2	1
NRF Client NF Management	1	2	1	2	1
Perf-Info	1	2	1	2	1
App-info	1	2	1	2	1

Table 2-6 (Cont.) Upgrade

Service	CPU		Memory (Gi)		Replica(s)
	Min	Max	Min	Max	Max
Query Service	1	2	1	2	1
CM Service	1	2	1	2	1
Config Server	1	2	1	2	1
Audit Service	1	2	1	2	1

2.2 Installation Sequence

This section describes preinstallation, installation, and postinstallation tasks for BSF.

2.2.1 Preinstallation Tasks

Before installing BSF, perform the tasks described in this section.

2.2.1.1 Verifying and Creating Namespace

This section explains how to verify and create a namespace in the system.

Note

This is a mandatory procedure, run this before proceeding further with the installation. The namespace created or verified in this procedure is an input for the next procedures.

To verify and create a namespace:

1. Run the following command to verify if the required namespace already exists in system:

```
kubectl get namespaces
```

In the output of the above command, if the namespace exists, continue with [Creating Service Account, Role, and RoleBinding](#).

2. If the required namespace is unavailable, create the namespace using the following command:

```
kubectl create namespace <required namespace>
```

Where, <required namespace> is the name of the namespace.

For example:

```
kubectl create namespace ocbsf
```

Sample output:

```
namespace/ocbsf created
```

Naming Convention for Namespaces

The namespace must meet the following requirements:

- The namespace should start and end with an alphanumeric character.
- contains 63 characters or less
- contains only alphanumeric characters or '-'

Note

It is recommended to avoid using the prefix kube- when creating namespace. The prefix is reserved for Kubernetes system namespaces.

2.2.1.2 Creating Service Account, Role, and RoleBinding

This section is optional and it describes how to manually create a service account, Role, and RoleBinding resources.

Note

The secret(s) should exist in the same namespace where BSF is getting deployed. This helps to bind the Kubernetes role with the given service account.

To create service account, role and rolebinding:

1. Create a Global Service Account.

create a YAML file `bsf-sample-serviceaccount-template.yaml` using the following sample code:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <helm-release>-serviceaccount
  namespace: <namespace>
```

Where,

<helm-release> is a name provided to identify the Helm deployment.

<namespace> is a name provided to identify the Kubernetes namespace of BSF. All the BSF microservices are deployed in this Kubernetes namespace.

2. Define role permissions using roles for the BSF namespace.

Create a YAML file `bsf-sample-role-template.yaml` using the following sample code:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: <helm-release>-role
```

```

    namespace: <namespace>
  rules:
  - apiGroups:
    - discovery.k8s.io
    resources:
    - endpointslices
    verbs:
    - get
    - watch
    - list
  - apiGroups:
    - ""
    resources:
    - services
    - configmaps
    - pods
    - secrets
    - endpoints
    - nodes
    - events
    - persistentvolumeclaims
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - apps
    resources:
    - deployments
    - statefulsets
    verbs:
    - get
    - watch
    - list
  - apiGroups:
    - autoscaling
    resources:
    - horizontalpodautoscalers
    verbs:
    - get
    - watch
    - list

```

3. To bind the role defined in the `bsf-sample-role-binding` template.yaml file with the service account, create a `bsf-sample-rolebinding-template.yaml` file using the following sample code:

```

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

```

```
subjects:
- kind: ServiceAccount
  name: <helm-release>-serviceaccount
  namespace: <namespace>
```

Note

If you are installing BSF 22.1.0 using CNE 22.2.0 or later versions, change the `apiVersion` from `rbac.authorization.k8s.io/v1beta1` to `rbac.authorization.k8s.io/v1`.

- Run the following commands to create resources:

```
kubectl -n <namespace> create -f bsf-sample-serviceaccount-template.yaml;
kubectl -n <namespace> create -f bsf-sample-role-template.yaml;
kubectl -n <namespace> create -f bsf-sample-rolebinding-template.yaml
```

Note

Once the global service account is added, you must add `globalServiceAccountName` in the `ocbsf_custom_values_26.1.201.yaml` file. Otherwise, installation may fail as a result of creating and deleting Custom Resource Definition (CRD).

Note

`PodSecurityPolicy` kind is required for Pod Security Policy service account. For more information, see *Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*.

2.2.1.3 Configuring cnDBTier

With `cnDBTier`, BSF facilitates automatic user creation with its pre-install hook. However, ensure that there is a privileged user on the NDB cluster, which has privileges similar to root user. You must have necessary permissions to allow connections from remote hosts.

Single Site Deployment

Perform the following steps on each of the SQL nodes.

- Log in to MySQL on each of the API nodes of `cnDBTier` to verify this.

```
mysql>select host from mysql.user where User='<privileged username>';
+-----+
| host |
+-----+
| %    |
+-----+
1 rowinset(0.00 sec)
```

2. If you do not see '%' in the output of the above query, modify this field to allow remote connections to root.

```
mysql>update mysql.user set host='%' where User='<privileged username>';
Query OK, 0rowsaffected (0.00 sec)
Rowsmatched: 1 Changed: 0 Warnings: 0
mysql> flush privileges;
Query OK, 0rowsaffected (0.06 sec)
```

Note

Perform this step on each SQL node.

Multisite Deployment

To configure cnDBTier in case of multisite deployment:

1. Update mysqld configuration in the cnDBTier custom-values.yaml file before installing or upgrading BSF.

```
global:
  ndbconfigurations:
    api
      auto_increment_increment: 3
      auto_increment_offset: 1
```

Note

- Set the auto_increment_increment parameter same as number of sites. For example: If the number of sites is 2, set its value as 2 and if the number of sites is 3, set its value as 3.
- Set the auto_increment_offset parameter as site ID. For example: The site ID for Site 1 is 1, Site 2 is 2, for Site 3 is 3, and so on.

2. If the fresh installation or upgrade of BSF on cnDBTier is not planned, then run the following command to edit the mysqldconfig configmap on all the cnDBTier sites.

```
kubectl edit configmap mysqldconfig <db-site-namespace>
```

For example:

```
kubectl edit configmap mysqldconfig -n site1
```

Note

Update the auto_increment_increment and auto_increment_offset values as mentioned in the previous step for all sites.

2.2.1.4 Configuring Multiple Site Deployment

In case of multiple site deployment of BSF, there is only one subscriber database which is used by each site and different configuration databases for each site, as each site has its own configuration. To have different configuration databases and same subscriber database, you need to create secrets accordingly. For more information about creating secrets, see [Configuring Kubernetes Secret for Accessing Database](#).

Note

In a BSF georedundant deployments, while adding a new BSF site ensure that its version is same as the other existing BSF site versions.

To configure multiple site deployment:

1. Configure `nfInstanceId` under the global section of the `ocbsf_custom_values_26.1.201.yaml` file differently for each BSF site deployed.

Note

Ensure that the `nfInstanceId` configuration in the global section is same as that in the `appProfile` section of `nrf-client`.

global:

```
# Unique ID to register to NRF, Should be configured differently on
multi site deployments for each BSF
nfInstanceId: &nfInsId 5a7bd676-ceed-44bb-95e0-f6a55a328b03
```

nrf-client:

```
configmapApplicationConfig:
  profile: |-
    appProfiles=[{"nfInstanceId":"5a7bd676-ceed-44bb-95e0-
f6a55a328b03","nfStatus":"REGISTERED","fqdn":"ocbsf-
ingressgateway.mybsf.svc.cluster.local","nfType":"BSF","allowedNfTypes":
["NRF"],"plmnList":
[{"mnc":"14","mcc":"310"}],"priority":10,"capacity":500,"load":0,"locality":
"bangalore","nfServices":[{"load":0,"scheme":"http","versions":
[{"apiFullVersion":"2.1.0.alpha-3","apiVersionInUri":"v1"}],"fqdn":"ocbsf-
ingressgateway.mybsf.svc.cluster.local","ipEndPoints":
[{"port":"80","ipv4Address":"10.0.0.0","transport":"TCP"}],"nfServiceStatus":
"REGISTERED","allowedNfTypes":
["NRF"],"serviceInstanceId":"547d42af-628a-4d5d-
a8bd-38c4ba672682","serviceName":"nbsf-group-id-
map","priority":10,"capacity":500},"udrInfo":
{"groupId":"bsf-1","externalGroupIdentifiersRanges":
[{"start":"10000000000","end":"20000000000"}],"supiRanges":
[{"start":"10000000000","end":"20000000000"}],"gpsiRanges":
[{"start":"10000000000","end":"20000000000"}],"heartBeatTimer":90,"nfServi
cePersistence":false,"nfProfileChangesSupportInd":false,"nfSetIdList":
["setxyz.bsfset.5gc.mnc012.mcc345"]}]]
```

2. Configure `fullnameOverride` under the `config-server` section to `<helm-release-name>-config-server`. It should be different for each site deployed.

```
config-server:
  fullnameOverride: ocudr1-config-server
```

3. Configure `fullnameOverride` under the `appinfo` section to `<helm-release-name>-app-info`. It should be different for each site deployed.

```
appinfo:
  fullnameOverride: ocudr1-app-info
```

4. For `cnDBTier` configurations in multiple site deployment, see [Configuring cnDBTier](#).

2.2.1.5 Creating Service Account, Role, and Role Binding for Helm Test

This section describes the procedure to create service account, role, and role binding resources for Helm Test.

! Important

The steps described in this section are optional and users may skip it in any of the following scenarios:

- If user wants service accounts to be created automatically at the time of deploying BSF.
- Global service account with associated role and role bindings is already configured or the user has any in-house procedure to create service accounts.

Creating Global Service Account

To create the global service account, create a YAML file `bsf-sample-helmtestserviceaccount-template.yaml` using the following sample code:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <helm-release>-helmtestserviceaccount
  namespace: <namespace>
```

Where,

`<helm-release>` is a name provided to identify the helm deployment.

`<namespace>` is a name provided to identify the Kubernetes namespace of BSF. All the BSF microservices are deployed in this Kubernetes namespace.

Define Role Permissions

To define permissions using roles for the BSF namespace, create a YAML file `bsf-sample-helmtestrole-template.yaml` using the following sample code:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

```
metadata:
  name: <helm-release>-helmtest-role
  namespace: <namespace>
rules:
  - apiGroups:
      - ""
    resources:
  - pods
    - persistentvolumeclaims
    - services
    - endpoints
    - configmaps
    - events
    - secrets
    - serviceaccounts
  verbs:
    - list
    - get
    - watch
  - apiGroups:
      - apps
    resources:
    - deployments
    - statefulsets
  verbs:
    - get
    - watch
    - list
  - apiGroups:
      - autoscaling
    resources:
    - horizontalpodautoscalers
  verbs:
    - get
    - watch
    - list
  - apiGroups:
      - policy
    resources:
    - poddisruptionbudgets
  verbs:
    - get
    - watch
    - list
  - apiGroups:
      - rbac.authorization.k8s.io
    resources:
    - roles
    - rolebindings
  verbs:
    - get
    - watch
    - list
```

Creating Helm Test Role Binding Template

To bind the role defined in the `bsf-sample-role-template.yaml` file with the service account, create a `bsf-sample-helmttestrolebinding-template.yaml` file using the following sample code:

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

Creating resources

Run the following commands to create resources:

```
kubectl -n <namespace> create -f bsf-sample-helmttestserviceaccount-
template.yaml;
kubectl -n <namespace> create -f bsf-sample-role-template.yaml;
kubectl -n <namespace> create -f bsf-sample-rolebinding-template.yaml
```

Note

Once the global service account is added, users must add `helmTestServiceAccountName` in the `ocbsf_custom_values_26.1.201.yaml` file. Otherwise, installation may fail as a result of creating and deleting Custom Resource Definition (CRD).

2.2.1.6 Configuring Database, Creating Users, and Granting Permissions

This section explains how database administrators can create users and database in a single and multisite deployment.

BSF has four databases (Provisional, State, Release, Leaderpod, and NRF Client Database) and two users (Application and Privileged).

Note

- Before running the procedure for georedundant sites, ensure that the cnDBTier for georedundant sites is already up and replication channels are enabled.
- While performing a fresh installation, if BSF release is already deployed, purge the deployment and remove the database and users that were used for the previous deployment. For uninstallation procedure, see [Uninstalling BSF](#).

Note

The following are the requirements to be followed while creating a database password:

- The recommended characters are only A-Z, a-z, 0-9, ., _, ~, and -.
- Any reserved characters for URLs such as /, :, @, (,), [,], &, #, =, ? and spaces should not be used.

BSF Databases

For BSF applications, four types of databases are required:

1. **Provisional Database:** Provisional Database contains configuration information. The same configuration must be done on each site by the operator. Both Privileged User and Application User have access to this database. In case of georedundant deployments, each site must have a unique Provisional Database. BSF sites can access only the information in their unique Provisional Database.

For example:

- For Site 1: ocbsf_config_server_site1
- For Site 2: ocbsf_config_server_site2
- For Site 3: ocbsf_config_server_site3

2. **State Database:** This database maintains the running state of BSF sites and has information of subscriptions, pending notification triggers, and availability data. It is replicated and the same configuration is maintained by all BSF georedundant sites. Both Privileged User and Application User have access to this database.

3. **Release Database:** This database maintains release version state, and it is used during upgrade and rollback scenarios. Only Privileged User has access to this database.

4. **Leaderpod Database:** This database is used to store leader and follower if PDB is enabled for microservices that require a single pod to be up in all the instances. The configuration of this database must be done on each site. In case of georedundant deployments, each site must have a unique Leaderpod database.

For example:

- For Site 1: ocbsf_leaderPodDb_site1
- For Site 2: ocbsf_leaderPodDb_site2
- For Site 3: ocbsf_leaderPodDb_site3

Note

This database is used only when `nrf-client-nfmanagement.enablePDBSupport` is set to `true` in the `ocbsf_custom_values_26.1.201.yaml`.

5. **NRF Client Database:** This database is used to support NRF Client features. Only Privileged User has access to this database and it is used only when the caching feature is enabled. In case of georedundant deployments, each site must have a unique NRF Client database and its configuration must be done on each site.

For example:

- For Site 1: `ocbsf_nrf_client_site1`
- For Site 2: `ocbsf_nrf_client_site2`
- For Site 3: `ocbsf_nrf_client_site3`

BSF Users

There are two types of BSF database users with different set of permissions:

1. **Privileged User:** This user has a complete set of permissions. This user can perform create, alter, or drop operations on tables to perform install, upgrade, rollback, or delete operations.

Note

In examples given in this document, Privileged User's username is 'bsfprivilegedusr' and password is 'bsfprivilegedpasswd'.

2. **Application User:** This user has a limited set of permissions and is used by BSF application to handle service operations. This user can insert, update, get, or remove the records. This user will not be able to create, alter, or drop the database or tables.

Note

In examples given in this document, Application User's username is 'bsfusr' and password is 'bsfpasswd'.

Before creating a user in multi site deployment the BINLOG must be disabled. The BINLOG must be enabled after the user creation.

Run the following command to disable the binary logging for the current session:

```
SET sql_log_bin = OFF;
```

Run the following command to enable the binary logging for the current session:

```
SET sql_log_bin = ON;
```

Default Databases

BSF microservices use the MySQL database to store the configuration and run time data.

Before deploying BSF, make sure that the MySQL user and databases are created.

Each microservice has a default database assigned to it.

The following table lists the default database names and applicable deployment modes for various databases that need to be configured while deploying BSF.

Table 2-7 Default Database Names for BSF Microservices

Service Name	Default Database Name	Database Type
Config Server	ocbsf_config_server	Provisional
CM Service	ocbsf_commonconfig ocbsf_cmService	Provisional
BSF Service	ocpm_bsf	State
Audit Service	ocbsf_audit_service	Provisional
NRF Client	ocbsf_nrf_client ocbsf_leaderPodDb	Provisional
Query Service	ocbsf_query	State
Perf Info Service	ocbsf_overload	Leaderpod

In addition, create the **ocbsf_release** (default name) database to store and manipulate the release versions of BSF services during the install, upgrade, and rollback procedure. This database name is specified in the **releaseDbName** parameter in the `ocbsf_custom_values_26.1.201.yaml` file.

2.2.1.6.1 Single Site

This section explains how a database administrator can create database and users for a single site deployment.

Configuring Database

Perform the following steps to configure MySQL database for different microservices:

1. Log in to the server where the SSH keys are stored and have permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL nodes.
3. Log in to the MySQL prompt using root permission, or log in as a user who has the permission to create users as per conditions explained in the next step.
Example:

```
mysql -h 127.0.0.1 -uroot -p
```

Note

This command varies between systems, path for MySQL binary, root user, and root password. After running this command, enter the password specific to the user mentioned in the command.

4. Run the following command to check if both the BSF users already exist:

```
SELECT User FROM mysql.user;
```

If the users already exist, go to the next step. Else, create the respective new user or users by following the steps below:

- Run the following command to create a new Privileged User:

```
CREATE USER '<BSF Privileged-User Name>'@'%' IDENTIFIED BY '<BSF Privileged-User Password>';
```

Example:

```
CREATE USER 'bsfprivilegedusr'@'%' IDENTIFIED BY 'bsfprivilegedpasswd';
```

- Run the following command to create a new Application User:

```
CREATE USER '<Application User Name>'@'%' IDENTIFIED BY '<APPLICATION Password>';
```

Example:

```
CREATE USER 'bsfusr'@'%' IDENTIFIED BY 'bsfpasswd';
```

5. Run the following command to check whether any of the BSF databases already exists:

```
show databases;
```

- a. If any of the previously configured database is already present, remove them. Otherwise, skip this step.

Run the following command to remove a preconfigured BSF database:

```
DROP DATABASE if exists <DB Name>;
```

Example:

```
DROP DATABASE if exists ocbsf_audit_service;
```

- b. Run the following command to create new BSF database if it does not exist, or after dropping an existing database:

```
CREATE DATABASE IF NOT EXISTS <DB Name> CHARACTER SET utf8;
```

For example:

```
CREATE DATABASE IF NOT EXISTS ocbsf_config_server CHARACTER SET utf8;  
CREATE DATABASE IF NOT EXISTS ocbsf_release CHARACTER SET utf8;  
CREATE DATABASE IF NOT EXISTS ocbsf_commonconfig CHARACTER SET utf8;  
CREATE DATABASE IF NOT EXISTS ocbsf_cmsservice CHARACTER SET utf8;  
CREATE DATABASE IF NOT EXISTS ocbsf_audit_service CHARACTER SET utf8;
```

```
CREATE DATABASE IF NOT EXISTS ocpm_bsf CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_nrf_client CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_leaderPodDb CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_query CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_overload CHARACTER SET utf8;
```

Note

Ensure that you use the same database names while creating database that you have used in the global parameters of ocbsf_custom_values_26.1.201.yaml files.

Following is an example of what are the names of the BSF database names configured in the ocbsf_custom_values_26.1.201.yaml file:

```
global:
  releaseDbName: &releaseDbName 'ocbsf_release'
  nrfClientDbName: 'ocbsf_nrf_client'
bsf-management-service:
  envMySQLDatabase: 'ocpm_bsf'
config-server:
  envMySQLDatabase: *configServerDB
cm-service:
  envMySQLDatabase: ocbsf_cmService
nrf-client-nfmanagement:
  dbConfig:
    leaderPodDbName: ocbsf_leaderPodDb
audit-service:
  envMySQLDatabase: ocbsf_audit_service
  recordsQueueCapacity: 100000
queryservice:
  envMySQLDatabase: ocbsf_query
```

Name of the config-server database is defined using *configServerDB variable. The actual values of this variable (config-server database name) must be defined under the **commonRef:** section.

```
commonRef: # References for common key-value pairs
  # Config Server DB reference
  - &configServerDB ocbsf_config_server
```

BSF follows the best database practices by having the idle connection timeout for client applications lesser than the idle connection timeout of the database server.

The default idle connection timeout value of BSF applications is 540 seconds or 9 minutes. This value remains changed.

Creating Users and Granting Permissions

Note

Creation of database is optional if grant is scoped to all database, that is, database name is not mentioned in grant command.

To create users and grant permissions, perform the following steps:

1. Run the following set of commands to grant all the necessary permissions:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, ALTER,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE ON <DB Name>.* TO `BSF
Privileged-User Name`@`%`;
```

In the following example, "bsfprivilegedusr" is used as username, "bsfprivilegedpasswd" is used as password. Here, all permissions are being granted to "bsfprivilegedusr".

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_config_server.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_release.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocpm_bsf.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_commonconfig.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_audit_service.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_cmsservice.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON
mysql.ndb_replication TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX,
ALTER ON ocbsf_nrf_client.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE TEMPORARY
TABLES, DELETE, UPDATE, EXECUTE ON ocbsf_leaderPodDb.* TO
'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_query.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_overload.* TO 'bsfprivilegedusr'@'%';
```

2. Run the following command to grant NDB_STORED_USER permissions to the Privileged User:

```
GRANT NDB_STORED_USER ON *.* TO 'bsfprivilegedusr'@'%';
```

3. Grant all the necessary permissions by running the following set of commands.

Note

The database name is specified in the **envMySQLDatabase** parameter for respective services in the ocbsf_custom_values_26.1.201.yaml file.

It is recommended to use a unique database name when there are multiple instances of BSF deployed in the network as they share the same data tier (MySQL cluster).

To grant permissions:

```
GRANT SELECT, INSERT, LOCK TABLES, DELETE, UPDATE, REFERENCES, EXECUTE ON
<DB Name>.* TO '<Application User Name>'@'%';
```

In the following example, "bsfusr" is used as username, "bsfpasswd" is used as password. Here, all permissions are being granted to "bsfusr".

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ocbsf_config_server.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE ON ocbsf_release.* TO 'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE ON ocpm_bsf.* TO 'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_commonconfig.* TO
'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_audit_service.* TO
'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_nrf_client.* TO
'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_cmsservice.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_query.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_overload.* TO
'bsfusr'@'%';
```

4. Run the following command to grant NDB_STORED_USER permissions to the Application User:

```
GRANT NDB_STORED_USER ON *.* TO 'bsfusr'@'%';
```

5. Run the following commands to verify that the privileged or application users have all the required permission:

```
show grants for username;
```

where username is the name of the privileged or application user.

Example:

```
show grants for bsfprivilegedusr;
show grants for bsfusr;
```

6. Run the following command to flush privileges:

```
FLUSH PRIVILEGES;
```

7. Exit from the database and logout from the MySQL node.

2.2.1.6.2 Multisite

This section explains how database administrator can create the databases and users for a multisite deployment.

For BSF georedundant deployment, listed databases names must be unique for each site. For the remaining databases, the database name must be same across all the sites.

It is recommended to use unique database names when multiple instances of BSF use and share a single cnDBTier (MySQL cluster) in the network. To maintain unique database names for all the NF instances in the network, a good practice is to add the deployment name of the BSF instance as a prefix or suffix to the database name. However, you can use any prefix or suffix to create the unique database name. For example, if the BSF deployment nInstance value is "site1" then the BSF Configuration service database can be named as "ocbsf_config_server_site1".

Note

Before running the procedure for georedundant sites, ensure that the cnDBTier for georedundant sites is up and replication channels are enabled.

Table 2-8 BSF Unique Databases names for two site and three site deployment

Two Site Database Names	Three Site Database Names
ocbsf_config_server_site1 ocbsf_config_server_site2	ocbsf_config_server_site1 ocbsf_config_server_site2 ocbsf_config_server_site3
ocbsf_cmsservice_site1 ocbsf_cmsservice_site2	ocbsf_cmsservice_site1 ocbsf_cmsservice_site2 ocbsf_cmsservice_site3
ocbsf_commonconfig_site1 ocbsf_commonconfig_site2	ocbsf_commonconfig_site1 ocbsf_commonconfig_site2 ocbsf_commonconfig_site3
ocbsf_leaderPodDb_site1 ocbsf_leaderPodDb_site2	ocbsf_leaderPodDb_site1 ocbsf_leaderPodDb_site2 ocbsf_leaderPodDb_site3
ocbsf_overload_site1 ocbsf_overload_site2	ocbsf_overload_site1 ocbsf_overload_site2 ocbsf_overload_site3
ocbsf_audit_service_site1 ocbsf_audit_service_site2	ocbsf_audit_service_site1 ocbsf_audit_service_site2 ocbsf_audit_service_site3
ocbsf_nrf_client_site1 ocbsf_nrf_client_site2	ocbsf_nrf_client_site1 ocbsf_nrf_client_site2 ocbsf_nrf_client_site3

Configuring Database

Perform the following steps to configure MySQL database for different microservices:

1. Log in to the server where the SSH keys are stored and have permission to access the SQL nodes of NDB cluster.
2. Connect to the SQL nodes.

3. Log in to the database either as a root user or as a user who has the permission to create users as per conditions explained in the next step.

Example:

```
mysql -h 127.0.0.1 -uroot -p
```

Note

This command varies between systems, path for MySQL binary, root user, and root password. After running this command, enter the password specific to the user mentioned in the command.

4. Run the following command to check if both the BSF users already exist:

```
SELECT User FROM mysql.user;
```

If the users already exist, go to the next step. Otherwise, create the respective new user or users by following the steps below:

- Run the following command to create a new Privileged User:

```
CREATE USER '<BSF Privileged-User Name>'@'%' IDENTIFIED BY '<BSF Privileged-User Password>';
```

Example:

```
CREATE USER 'bsfprivilegedusr'@'%' IDENTIFIED BY 'bsfprivilegedpasswd';
```

- Run the following command to create a new Application User:

```
CREATE USER '<Application User Name>'@'%' IDENTIFIED BY '<APPLICATION Password>';
```

Example:

```
CREATE USER 'bsfusr'@'%' IDENTIFIED BY 'bsfpasswd';
```

Note

You must create both the users on all the SQL nodes for all georedundant sites.

5. Update the cnDBTier yaml file to add the databases to the existing replication channel.

- Add <database name> database to channelgroupid: <default group ID> inside binlogdodb section.

Where:

- <database name> is the name of the database.
- <default group ID> is the replication group that contains all the databases other than the high update volume databases, which are assigned to dedicated

replication groups. The <default group ID> can be different for different use cases.

Add all the required databases to cnDBTier yaml file as explained above.

- Add all the databases to the other replication channels inside binlogignoredb section.

Note

The cnDBTier yaml file must be updated at each site prior to the Helm upgrade on the associated site.

6. Perform Helm upgrade on all the sites.
7. Run the following command to check whether any of the BSF databases already exists:

```
show databases;
```

- a. If any of the previously configured database is already present, remove them. Otherwise, skip this step.

Caution

In case you have georedundant sites configured, removal of the database from any one of the SQL nodes of any cluster will remove the database from all georedundant sites.

Run the following command to remove a preconfigured BSF database:

```
DROP DATABASE if exists <DB Name>;
```

Example:

```
DROP DATABASE if exists ocbsf_audit_service;
```

- b. Run the following command to create new BSF database if it does not exist, or after dropping an existing database:

```
CREATE DATABASE IF NOT EXISTS <DB Name> CHARACTER SET utf8;
```

For example: Sample illustration for creating all database required for BSF installation in site1.

```
CREATE DATABASE IF NOT EXISTS ocbsf_config_server_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_commonconfig_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_cmsservice_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_audit_service_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_nrf_client_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_leaderPodDb_site1 CHARACTER SET
```

```
utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_query CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_overload_site1 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocpm_bsf CHARACTER SET utf8;
```

For example: Sample illustration for creating all database required for BSF installation in site2.

```
CREATE DATABASE IF NOT EXISTS ocbsf_config_server_site2 CHARACTER SET
utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_commonconfig_site2 CHARACTER SET
utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_cmsservice_site2 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_audit_service_site2 CHARACTER SET
utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_nrf_client_site2 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_leaderPodDb_site2 CHARACTER SET
utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_query CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocbsf_overload_site2 CHARACTER SET utf8;
CREATE DATABASE IF NOT EXISTS ocpm_bsf CHARACTER SET utf8;
```

Note

Ensure that you use the same database names while creating database that you have used in the global parameters of `ocbsf_custom_values_26.1.201.yaml` files.

BSF follows the best database practices by having the idle connection timeout for client applications lesser than the idle connection timeout of the database server.

The default idle connection timeout value of BSF applications is 540 seconds or 9 minutes. This value remains changed.

Granting Permissions to Users on the Database

Note

- Run this step on all the SQL nodes for each BSF standalone site in a georedundant deployment.
- Creation of database is optional if grant is scoped to all databases, that is, database name is not mentioned in grant command.

1. Run the following command to grant Privileged User permission on all BSF Databases:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, ALTER,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE ON <DB Name>_<site_name>.*
TO `<BSF Privileged-User Name>`@`%`;
```

In the following example, "bsfprivilegedusr" is used as username, "bsfprivilegedpasswd" is used as password. Here, all permissions are being granted to "bsfprivilegedusr".

Example for site1:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_config_server_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_commonconfig_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_audit_service_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_cmsservice_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON
mysql.ndb_replication TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX,
ALTER ON ocbsf_nrf_client_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE TEMPORARY
TABLES, DELETE, UPDATE, EXECUTE ON ocbsf_leaderPodDb_site1.* TO
'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_query.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_overload_site1.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocpm_bsf.* TO 'bsfprivilegedusr'@'%';
```

Example for site2:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_config_server_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_commonconfig_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_audit_service_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_cmsservice_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON
mysql.ndb_replication TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX,
ALTER ON ocbsf_nrf_client_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, CREATE, ALTER, DROP, LOCK TABLES, CREATE TEMPORARY
TABLES, DELETE, UPDATE, EXECUTE ON ocbsf_leaderPodDb_site2.* TO
'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_query.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocbsf_overload_site2.* TO 'bsfprivilegedusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, REFERENCES,
INDEX ON ocpm_bsf.* TO 'bsfprivilegedusr'@'%';
```

2. Run the following command to grant NDB_STORED_USER permissions to the Privileged User:

```
GRANT NDB_STORED_USER ON *.* TO 'bsfprivilegedusr'@'%';
```

3. Run the following command to grant Application User permission on all BSF Databases:

```
GRANT SELECT, INSERT, LOCK TABLES, DELETE, UPDATE, REFERENCES, EXECUTE ON
<DB Name>_<site_name>.* TO '<Application User Name>'@'%';
```

In the following example, "bsfusr" is used as username, "bsfpasswd" is used as password. Here, all permissions are being granted to "bsfusr".

For example in BSF site1:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ocbsf_config_server_site1.* TO
'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_commonconfig_site1.*
TO 'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON
ocbsf_audit_service_site1.* TO 'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_nrf_client_site1.*
TO 'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_cmsservice_site1.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_query.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_overload_site1.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocpm_bsf.* TO 'bsfusr'@'%';
```

Example for site2:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ocbsf_config_server_site2.* TO
'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_commonconfig_site2.*
TO 'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON
ocbsf_audit_service_site2.* TO 'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_nrf_client_site2.*
TO 'bsfusr'@'%';
GRANT CREATE, SELECT, INSERT, UPDATE, DELETE ON ocbsf_cmsservice_site2.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_query.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocbsf_overload_site2.* TO
'bsfusr'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON ocpm_bsf.* TO 'bsfusr'@'%';
```

4. Run the following command to grant NDB_STORED_USER permissions to the Application User:

```
GRANT NDB_STORED_USER ON *.* TO 'bsfusr'@'%';
```

5. Run the following command to verify that the privileged or application users have all the required permissions:

```
show grants for username;
```

where username is the name of the privileged or application user.

For example:

```
show grants for bsfprivilegedusr;  
show grants for bsfusr;
```

6. Run the following command to flush privileges:

```
FLUSH PRIVILEGES;
```

7. Exit from MySQL prompt and SQL nodes.

2.2.1.7 Configuring Kubernetes Secret for Accessing Database

This section explains how to configure Kubernetes secrets for accessing BSF database.

2.2.1.7.1 Creating and Updating Secret for Privileged Database User

This section explains how to create and update Kubernetes secret for Privileged User to access the database.

1. Run the following command to create Kubernetes secret:

```
kubectl create secret generic <Privileged User secret name> --from-  
literal=mysql-username=<Privileged Mysql database username> --from-  
literal=mysql-password=<Privileged Mysql User database password> -n  
<Namespace>
```

Where,

<Privileged User secret name> is the secret name of the Privileged User.

<Privileged MySQL database username> is the username of the Privileged User.

<Privileged MySQL User database password> is the password of the Privileged User.

<Namespace> is the namespace of BSF deployment.

Note

Note down the command used during the creation of Kubernetes secret. This command is used for updating the secrets in future.

For example:

```
kubectl create secret generic ocbsf-privileged-db-pass --from-  
literal=mysql-username=bsfprivilegedusr --from-literal=mysql-  
password=bsfprivilegedpasswd -n ocbsf
```

2. Run the following command to verify the secret created:

```
kubectl describe secret <Privileged User secret name> -n <Namespace>
```

Where,

<Privileged User secret name> is the secret name of the database.

<Namespace> is the namespace of BSF deployment.

For example:

```
kubectl describe secret ocbsf-privileged-db-pass -n ocbsf
```

Sample output:

```
Name: ocbsf-privileged-db-pass
```

```
Namespace: ocbsf
```

```
Labels: <none>
```

```
Annotations: <none>
```

```
Type: Opaque
```

```
Data
```

```
====
```

```
mysql-password: 10 bytes
```

```
mysql-username: 17 bytes
```

3. Update the command used in step 1 with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of BSF deployment>". After the update is performed, use the following command:

```
kubectl create secret generic <Privileged User secret name> --from-literal=dbUsername=<Privileged MySQL database username> --from-literal=dbPassword=<Privileged MySQL database password> --dry-run -o yaml -n <Namespace> | kubectl replace -f - -n <Namespace>
```

Where,

<Privileged User secret name> is the secret name of the Privileged User.

<Privileged MySQL database username> is the username of the Privileged User.

<Privileged MySQL User database password> is the password of the Privileged User.

<Namespace> is the namespace of BSF deployment.

4. Run the updated command. The following message is displayed:
secret/<Privileged User secret name> replaced

Where,

<Privileged User secret name> is the updated secret name of the Privileged User.

2.2.1.7.2 Creating and Updating Secret for Application Database User

This section explains how to create and update Kubernetes secret for application user to access the database.

1. Run the following command to create Kubernetes secret:

```
kubectl create secret generic <Application User secret name> --from-literal=mysql-username=<Application MySQL Database Username> --from-
```

```
literal=mysql-password=<Application MySQL User database password> -n  
<Namespace>
```

Where,

<Application User secret name> is the secret name of the Application User.

<Application MySQL database username> is the username of the Application User.

<Application MySQL User database password> is the password of the Application User.

<Namespace> is the namespace of BSF deployment.

Note

Note down the command used during the creation of Kubernetes secret. This command is used for updating the secrets in future.

For example:

```
kubectl create secret generic ocbsf-db-pass --from-literal=mysql-  
username=bsfusr --from-literal=mysql-password=bsfpasswd -n ocbsf
```

2. Run the following command to verify the secret created:

```
kubectl describe secret <Application User secret name> -n <Namespace>
```

Where,

<Application User secret name> is the secret name of the database.

<Namespace> is the namespace of BSF deployment.

For example:

```
kubectl describe secret ocbsf-db-pass -n ocbsf
```

Sample output:

```
Name: ocbsf-db-pass  
Namespace: ocbsf  
Labels: <none>  
Annotations: <none>
```

Type: Opaque

Data

====

```
mysql-password: 10 bytes  
mysql-username: 17 bytes
```

3. Update the command used in step 1 with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of BSF deployment>". After the update is performed, use the following command:

```
kubectl create secret generic <Application User secret name> --from-  
literal=dbUsername=<Application MySQL database username> --from-  
literal=dbPassword=<Application Mysql database password> --dry-run -o yaml  
-n <Namespace> | kubectl replace -f - -n <Namespace>
```

Where,

<Application User secret name> is the secret name of the Application User.

<Application MySQL database username> is the username of the Application User.

<Application MySQL User database password> is the password of the Application User.

<Namespace> is the namespace of BSF deployment.

4. Run the updated command. The following message is appears:
secret/<Application User secret name> replaced

Where,

<Application User secret name> is the updated secret name of the Application User.

2.2.1.7.3 Creating Secret for Support of TLS in Diameter Gateway

This section explains how to create Kubernetes secret to store private key, public key, and trust chain certificates to support TLS in Diameter Gateway.

1. Run the following command to create Kubernetes secret:

```
kubectl create secret generic <TLS_SECRET_NAME> --from-  
file=<TLS_RSA_PRIVATE_KEY_FILENAME/TLS_ECDSA_PRIVATE_KEY_FILENAME> --from-  
file=<TLS_CA_BUNDLE_FILENAME> --from-file=<TLS_RSA_CERTIFICATE_FILENAME/  
TLS_ECDSA_CERTIFICATE_FILENAME> -n <Namespace of OCCNP deployment>.
```

For example:

```
kubectl create secret generic dgw-tls-secret --from-file=dgw-key.pem --  
from-file=ca-cert.cer --from-file=dgw-cert.crt -n occnp
```

Where,

dgw-key.pem is the private Key of diam-gateway (either generated by RSA or ECDSA).

dgw-cert.crt is the public Key certificate of diam-gateway (either generated by RSA or ECDSA).

ca-cert.cer is the trust Chain Certificate file, either an Intermediate CA or Root CA.

dgw-tls-secret is the default name of the secret.

2.2.1.8 Configuring Secrets for Enabling HTTPS

This section explains the steps to create and update the Kubernetes secret and enable HTTPS at Ingress Gateway.

This step is optional. It is required only when SSL settings need to be enabled on Ingress Gateway and Egress Gateway microservices of BSF.

2.2.1.8.1 Configuring HTTPS at Ingress Gateway

This section explains the steps to configure secrets for enabling HTTPS in Ingress Gateway. This procedure must be performed before deploying CNC BSF.

Note

The passwords for TrustStore and KeyStore are stored in respective password files mentioned below.

To create Kubernetes secret for HTTPS, following files are required:

- ECDSA private key and CA signed certificate of OCBSF, if initialAlgorithm is ES256
- RSA private key and CA signed certificate of OCBSF, if initialAlgorithm is RS256
- TrustStore password file
- KeyStore password file
- Trust Chain Certificate file, either an Intermediate CA or Root CA,

Note

Creation process for private keys, certificates and passwords is based on discretion of user or operator.

2.2.1.8.1.1 Creating Secrets for Enabling HTTPS in Ingress Gateway

This section provides the steps to create secrets for enabling HTTPS in ingress gateway. Perform this procedure before deploying BSF.

1. Run the following command to create secret:

```
$ kubectl create secret generic <ocingress-secret-name> --from-  
file=<ssl_ecdsa_private_key.pem> --from-file=<rsa_private_key_pkcs1.pem> --  
from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-  
file=<caroot.cer> --from-file=<ssl_rsa_certificate.crt> --from-  
file=<ssl_ecdsa_certificate.crt> -n <Namespace of OCBSF deployment>
```

Note

Note down the command used during the creation of the secret. Use the command for updating the secrets in future.

Example: The names used below are same as provided in ocbf_custom_values_26.1.201.yaml in BSF deployment.

```
$ kubectl create secret generic ocingress-secret --from-  
file=ssl_ecdsa_private_key.pem --from-file=rsa_private_key_pkcs1.pem --
```

```
from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-  
file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-  
file=ssl_ecdsa_certificate.crt -n ocbsf
```

Note

It is recommended to use the same secret name as mentioned in the example. In case you change `<ocingress-secret-name>`, then update the `k8SecretName` parameter under `ingressgateway` attributes section in the `ocbsf_custom_values_26.1.201.yaml` file.

2. Run the following command to verify the secret created:

```
$ kubectl describe secret <ocingress-secret-name> -n <Namespace of OCBSF  
deployment>
```

Example:

```
$ kubectl describe secret ocingress-secret -n ocbsf
```

2.2.1.8.1.2 Updating Secrets for Enabling HTTPS in Ingress Gateway

This section explains how to update the secrets.

1. Copy the exact command used in the section during creation of secret.
2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of OCBSF deployment>".
The updated command is as follows:

```
$ kubectl create secret generic <ocingress-secret-name> --from-  
file=<ssl_ecdsa_private_key.pem> --from-file=<rsa_private_key_pkcs1.pem> --  
from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-  
file=<caroot.cer> --from-file=<ssl_rsa_certificate.crt> --from-  
file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of OCBSF  
deployment> | kubectl replace -f - -n <Namespace of OCBSF deployment>
```

Example:

```
$ kubectl create secret generic ocingress-secret --from-  
file=ssl_ecdsa_private_key.pem --from-file=rsa_private_key_pkcs1.pem --  
from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-  
file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-  
file=ssl_ecdsa_certificate.crt --dry-run -o yaml -n ocbsf | kubectl  
replace -f - -n ocbsf
```

Note

The names used in the aforementioned command must be as same as the names provided in the `ocbsf-26.1.201 ocbsf_custom_values_26.1.201.yaml` in OCCNP deployment.

3. Run the updated command.
After the secret update is complete, the following message appears:

```
secret/<ocingress-secret> replaced
```

2.2.1.8.1.3 Enabling HTTPS at Ingress Gateway

This step is optional. It is required only when SSL settings needs to be enabled on Ingress Gateway microservice of OCBSF.

1. Enable enableIncomingHttps parameter under Ingress Gateway Global Parameters section in the ocbsf_custom_values_26.1.201.yaml file. For more information about enableIncomingHttps parameter, see under global parameters section of the ocbsf_custom_values_26.1.201.yaml file. section.
2. Configure the following details in the ssl section under ingressgateway attributes, in case you have changed the attributes while creating secret:
 - Kubernetes namespace
 - Kubernetes secret name holding the certificate details
 - Certificate information

```
ingress-gateway:
  # ---- HTTPS Configuration - BEGIN ----
  enableIncomingHttps: false

  service:
    ssl:
      privateKey:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      certificate:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
      rsa:
        fileName: ocregress.cer
      caBundle:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: caroot.cer
      keyStorePassword:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: key.txt
      trustStorePassword:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: trust.txt
```

3. Save the ocbsf_custom_values_26.1.201.yaml file.

2.2.1.8.2 Configuring HTTPS at Egress Gateway

This section explains the steps to configure secrets for enabling HTTPS in Egress Gateway. This procedure must be performed before deploying OCBSF.

Note

The passwords for TrustStore and KeyStore are stored in respective password files mentioned below.

To create Kubernetes secret for HTTPS, following files are required:

- ECDSA private key and CA signed certificate of OCBSF, if initialAlgorithm is ES256
- RSA private key and CA signed certificate of OCBSF, if initialAlgorithm is RS256
- TrustStore password file
- KeyStore password file
- Trust Chain Certificate file, either an Intermediate CA or Root CA,

Note

Creation process for private keys, certificates and passwords is based on discretion of user or operator.

2.2.1.8.2.1 Creating Secrets for Enabling HTTPS in Egress Gateway

This section provides information about how to create secret for HTTPS related details. Perform this procedure before enabling HTTPS in OCBSF Egress Gateway.

1. Run the following command to create secret.

```
$ kubectl create secret generic <ocgress-secret-name> --from-  
file=<ssl_ecdsa_private_key.pem> --from-file=<ssl_rsa_private_key.pem> --  
from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-  
file=<ssl_cabundle.crt> --from-file=<ssl_rsa_certificate.crt> --from-  
file=<ssl_ecdsa_certificate.crt> -n <Namespace of OCBSF deployment>
```

Note

Note down the command used during the creation of the secret. Use the command for updating the secrets in future.

Example:

```
$ kubectl create secret generic ocgress-secret --from-  
file=ssl_ecdsa_private_key.pem --from-file=ssl_rsa_private_key.pem --from-  
file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-  
file=ssl_cabundle.crt --from-file=ssl_rsa_certificate.crt --from-  
file=ssl_ecdsa_certificate.crt -n ocbsf
```

Note

It is recommended to use the same secret name as mentioned in the example. In case you change <ocgress-secret-name>, then update the k8SecretName parameter under egressgateway attributes section in the ocbsf_custom_values_26.1.201.yaml file.

2. Run the following command to verify the secret created:

```
$ kubectl describe secret <ocgress-secret-name> -n <Namespace of OCBSF deployment>
```

Example:

```
$ kubectl describe secret ocgress-secret -n ocbsf
```

2.2.1.8.2.2 Updating Secrets for Enabling HTTPS in Egress Gateway

This section explains how to update the secret with related details.

1. Copy the exact command used in the section during creation of secret.
2. Update the same command with string "--dry-run -o yaml" and "kubectl replace -f - -n <Namespace of OCBSF deployment>".
The updated command is as follows:

```
kubectl create secret generic <ocgress-secret-name> --from-file=<ssl_ecdsa_private_key.pem> --from-file=<ssl_rsa_private_key.pem> --from-file=<ssl_truststore.txt> --from-file=<ssl_keystore.txt> --from-file=<ssl_cabundle.crt> --from-file=<ssl_rsa_certificate.crt> --from-file=<ssl_ecdsa_certificate.crt> --dry-run -o yaml -n <Namespace of OCBSF Egress Gateway secret> | kubectl replace -f - -n <Namespace of OCBSF deployment>
```

Example:

```
$ kubectl create secret generic egress-secret --from-file=ssl_ecdsa_private_key.pem --from-file=rsa_private_key_pkcs1.pem --from-file=ssl_truststore.txt --from-file=ssl_keystore.txt --from-file=caroot.cer --from-file=ssl_rsa_certificate.crt --from-file=ssl_ecdsa_certificate.crt --dry-run -o yaml -n ocbsf | kubectl replace -f - -n ocbsf
```

Note

The names used in the aforementioned command must be as same as the names provided in the ocbsf_custom_values_26.1.201.yaml in OCBSF deployment.

3. Run the updated command.
After the secret update is complete, the following message appears:

```
secret/<ocingress-secret> replaced
```

2.2.1.8.2.3 Enabling HTTPS at Egress Gateway

This step is optional. It is required only when SSL settings needs to be enabled on Egress Gateway microservice of OCBSF.

1. Enable `enableOutgoingHttps` parameter under `egressgateway` attributes section in the `ocbsf_custom_values_26.1.201.yaml` file. For more information about `enableOutgoingHttps` parameter, see the Egress Gateway section.
2. Configure the following details in the `ssl` section under `egressgateway` attributes, in case you have changed the attributes while creating secret:
 - Kubernetes namespace
 - Kubernetes secret name holding the certificate details
 - Certificate information

```
egress-gateway:
  #Enabling it for egress https requests
  enableOutgoingHttps: false

service:
  ssl:
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8NameSpace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      ecdsa:
        fileName: ssl_ecdsa_private_key.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8NameSpace: ocbsf
      rsa:
        fileName: ocegress.cer
      ecdsa:
        fileName: ssl_ecdsa_certificate.crt
    caBundle:
      k8SecretName: ocbsf-gateway-secret
      k8NameSpace: ocbsf
      fileName: caroot.cer
    keyStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8NameSpace: ocbsf
      fileName: key.txt
    trustStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8NameSpace: ocbsf
      fileName: trust.txt
```

3. Save the `ocbsf_custom_values_26.1.201.yaml` file.

2.2.1.9 Configuring Secret for Enabling Access Token Validation

This section explains how to configure a secret for enabling access token service.

2.2.1.9.1 Generating KeyPairs for NRF Instances

! Important

It is at the user's discretion to create the private keys and certificates, and it is not in the scope of BSF. This section lists only samples to create KeyPairs.

Using the Openssl tool, you can generate KeyPairs for each of the NRF instances. The commands to generate the KeyPairs are as follows:

i Note

Here, it is assumed that there are only two NRF instances with the the following instance IDs:

- NRF Instance 1: 664b344e-7429-4c8f-a5d2-e7dfaaaba407
- NRF Instance 2: 601aed2c-e314-46a7-a3e6-f18ca02faacc

Example Command to generate KeyPair for NRF Instance 1

Generate a 2048-bit RSA private key

```
openssl genrsa -out private_key.pem 2048
```

Convert private Key to PKCS#8 format (so Java can read it)

```
openssl pkcs8 -topk8 -inform PEM -outform PEM -in private_key.pem -out  
private_key_pkcs.der -nocrypt
```

Output public key portion in PEM format (so Java can read it)

```
openssl rsa -in private_key.pem -pubout -outform PEM -out public_key.pem
```

Create reqs.conf and place the required content for NRF certificate

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
C = IN
ST = BLR
L = TempleTerrace
O = Personal
CN = nnrf-001.tmtrflaa.5gc.tmp.com
[v3_req]
basicConstraints = CA:FALSE
keyUsage = digitalSignature, dataEncipherment
subjectAltName = DNS:nnrf-001.tmtrflaa.5gc.tmp.com
#subjectAltName = URI:UUID:6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c
#subjectAltName = otherName:UTF8:NRF
```

Output ECSDA private key portion in PEM format and corresponding NRF certificate in {nrfInstanceId}_ES256.crt file

```
openssl req -x509 -new -out {nrfInstanceId}_ES256.crt -newkey ec:<(openssl
ecparam -name secp521r1) -nodes -sha256 -keyout ecdsa_private_key.key -config
reqs.conf
```

#Replace the place holder "{nrfInstanceId}" with NRF Instance 1's UUID while running the command.

Example below

```
openssl req -x509 -new -out 664b344e-7429-4c8f-a5d2-e7dfaaaba407_ES256.crt -
newkey ec:<(openssl ecparam -name secp521r1) -nodes -sha256 -keyout
ecdsa_private_key.key -config reqs.conf
```

The output is a set of Private Key and NRF Certificate similar to the following:

NRF1 (Private key: ecdsa_private_key.key, NRF Public Certificate: 664b344e-7429-4c8f-a5d2-e7dfaaaba407_ES256.crt)

Example Command to generate KeyPair for NRF Instance 2

Generate a 2048-bit RSA private key

```
openssl genrsa -out private_key.pem 2048
Convert private Key to PKCS#8 format (so Java can read it)
```

```
openssl pkcs8 -topk8 -inform PEM -outform PEM -in private_key.pem -out
private_key_pkcs.der -nocrypt
Output public key portion in PEM format (so Java can read it)
```

```
openssl rsa -in private_key.pem -pubout -outform PEM -out public_key.pem
```

Create reqs.conf and place the required content for NRF certificate

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
C = IN
ST = BLR
L = TempleTerrace
O = Personal
CN = nnrf-001.tmtrflaa.5gc.tmp.com
[v3_req]
basicConstraints = CA:FALSE
keyUsage = digitalSignature, dataEncipherment
subjectAltName = DNS:nnrf-001.tmtrflaa.5gc.tmp.com
#subjectAltName = URI:UUID:6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c
#subjectAltName = otherName:UTF8:NRF
```

Output ECSDA private key portion in PEM format and corresponding NRF certificate in {nrfInstanceId}_ES256.crt file

```
openssl req -x509 -new -out {nrfInstanceId}_ES256.crt -newkey ec:<(openssl
ecparam -name prime256v1) -nodes -sha256 -keyout ecdsa_private_key.key -
config reqs.conf
```

#Replace the place holder "{nrfInstanceId}" with NRF Instance 2's UUID while

running the command.

Example below

```
openssl req -x509 -new -out 601aed2c-e314-46a7-a3e6-f18ca02faacc_ES256.crt -  
newkey ec:<(openssl ecparam -name prime256v1) -nodes -sha256 -keyout  
ecdsa_private_key.key -config reqs.conf
```

The output is a set of Private Key and NRF Certificate similar to the following:

NRF2 (Private key: ecdsa_private_key.key, PublicCertificate: 601aed2c-e314-46a7-a3e6-f18ca02faacc_ES256.crt)

2.2.1.9.2 Enabling and Configuring Access Token

Enabling and Configuring Access Token

To enable access token validation, configure both Helm-based and REST-based configurations on Ingress Gateway.

Configuration using Helm:

For Helm-based configuration, perform the following steps:

1. Create a Namespace for Secrets. The namespace is used as an input to create Kubernetes secret for private keys and public certificates. Create a namespace using the following command:

```
kubectl create namespace <required namespace>
```

Where,

<required namespace> is the name of the namespace.

For example, the following command creates the namespace, ocbf:

```
kubectl create namespace ocbf
```

2. Create Kubernetes Secret for NRF Public Key. To create a secret using the Public keys of the NRF instances, run the following command:

```
kubectl create secret generic <secret-name> --from-file=<filename.crt> -n  
<Namespace>
```

Where,

<secret-name> is the secret name.

<Namespace> is the BSF namespace.

<filename.crt> is the public key certificate and we can have any number of certificates in the secret.

For example:

```
kubectl create secret generic nrftpublickeysecret --from-file=./  
664b344e-7429-4c8fa5d2-e7dffaaba407_ES256.crt --from-file=./601aed2c-  
e314-46a7-a3e6-f18ca02faacc_ES256.crt -n ocbf
```

Note

In the above command:

- nrftpublickeysecret is the secret name
- ocbsf is the namespace
- .crt files is the public key certificates

3. Enable Access token using Helm Configuration by setting the Ingress Gateway parameter `oauthValidatorEnabled` parameter value to `true`.
Further, configure the secret and namespace on Ingress Gateway in the OAUTH CONFIGURATION section of the custom-value.yaml file.

The following is a sample Helm configuration. For more information on parameters and their supported values, see [OAUTH Configuration](#).

```
# ----OAUTH CONFIGURATION - BEGIN ----
oauthValidatorEnabled: true
nfInstanceId: 6faf1bbc-6e4a-4454-a507-a14ef8e1bc11
allowedClockSkewSeconds: 0
nrftPublicKeyKubeSecret: 'nrftpublickeysecret' //SECRET NAME
nrftPublicKeyKubeNamespace: 'ocbsf' //NAME SPACE of BSF
validationType: strict
producerPlmnMNC: 123
producerPlmnMCC: 456
nfType: BSF
```

Configuration using REST API:

After Helm configuration, send the REST requests to Ingress Gateway to use configured public key certificates. Using REST-based configuration, you can distinguish between the certificates configured on different NRFs and can use these certificates to validate the token received from a specific NRF.

Following are the 3 types of OAuth Validation Mode:

- **INSTANCEID_ONLY (Default):** Ingress Gateway validates access token based on public keys indexed with NRF Instance ID in the issuer field.
- **KID_ONLY:** Ingress Gateway validates access token based on public keys indexed with key-id only.
- **KID_PREFERRED:** Ingress Gateway validates access token based on public keys indexed with Key-ID. If Key-ID is not FOUND in Access token, Ingress Gateway attempts token validation using public keys indexed with NRF instance ID in the issuer field.

Table 2-9 Configuring oauth Validator

URI	Operation
/{nfType}/nf-common-component/v1/{serviceName}/ oauthvalidatorconfiguration For example: http://10.75.152.236:8000/bsf/nf-common-component/v1/igw/ oAuthValidatorConfiguration	PUT

Sample JSON:

```

"oAuthValidatorConfiguration": {
  "type": "object",
  "description": "Validator configurations for oAuth",
  "properties": {
    "keyIdList": {
      "type": ["array", "null"],
      "uniqueItems": true,
      "maxItems": 150,
      "description": "Array containing KID based configuration",
      "items": {
        "anyOf": [
          {
            "type": "object",
            "properties": {
              "keyId": {"type": "string", "minLength": 1, "maxLength":
36, "pattern": "[a-zA-Z0-9]"},
              "kSecretName": {"type": "string"},
              "certName": {"type": "string"},
              "certAlgorithm": {"type": "string"}
            },
            "required": [
              "keyId",
              "kSecretName",
              "certName",
              "certAlgorithm"
            ]
          }
        ]
      }
    },
    "instanceIdList": {
      "type": ["array", "null"],
      "uniqueItems": true,
      "maxItems": 150,
      "description": "Array containing Instance Id based configuration",
      "items": {

```

```

    "anyOf": [
      {
        "type": "object",
        "properties": {
          "instanceId": {"type": "string"},
          "kSecretName": {"type": "string"},
          "certName": {"type": "string"},
          "certAlgorithm": {"type": "string"}
        },
        "required": [
          "instanceId",
          "kSecretName",
          "certName",
          "certAlgorithm"
        ]
      }
    ]
  },
  "oauthValidationMode": {
    "type": "string",
    "enum": [
      "KID_ONLY",
      "INSTANCEID_ONLY",
      "KID_PREFERRED"
    ],
    "description": "Mode of validation"
  }
}

```

Validating OAuth Token

The following Curl command sends a request to create PCF Bindings with valid oAuth header:

```
curl -X POST --http2-prior-knowledge -i "http://10.75.233.75:32564/nbsf-
management/v1/pcfBindings" -H "Content-Type: application/json" -H
Authorization:'Bearer
eyJ0eXAiOiJKV1QiLCJrawQiOiI2MDFhZWQyYy1lMzE0LTQ2YTctYTlNlNi1mMTJhYyZmFheGhiLC
JhbGciOiJpFUFzI1NiJ9.eyJpc3MiOiI2NjRiMzQ0ZS03NDI5LTRjOGYtYTVkMi1lN2RmYWZhYmE0MDc
iLCJzdWIiOiJmZTdKOTkyYi0wNTQxLTRjN2Q2YWI4NC1jNmQ3MGlxYjAxYjEiLCJhdWQiOiJTTUYiL
CJzY29wZSI6Im5zbWYtcGRlc2Vzc2lubiIsImV4cCI6MTYxNzM1NzkzN30uOGAYtR3FnD33xOCmtUP
KBEA5RMTNvkfDqAK46ZEnnZvgN5Cyfgv1r85Zzdpo2lNISADBgDumD_m5xHJF8baNJQ'-d '{
  "supi": "imsi-310410000000015",
  "gpsi": "5084943708",
  "ipv4Addr": "10.10.10.10",
  "dnn": "internet",
  "pcfFqdn": "pcf-smsservice.oracle.com",
  "pcfDiamHost": "pcf-smsservice.oracle.com",
  "pcfDiamRealm": "oracle.com",
  "snssai": {
    "sst": 11,
    "sd": "abc123"
  }
}
```

2.2.1.10 Configuring BSF to Support Aspen Service Mesh

BSF leverages the Platform Service Mesh (for example, Aspen Service Mesh) for all internal and external Transport Layer Security (TLS) communication. The service mesh integration provides inter-NF communication and allows API gateway co-working with service mesh. The servicemesh integration supports the services by deploying a special sidecar proxy in each pod to intercept all network communications between microservices.

Supported ASM versions: 1.11.x, 1.14.x and 1.21.x

For ASM installation and configuration, see official Aspen Service Mesh website for details.

The Aspen Service Mesh (ASM) configurations are classified into:

- **Control Plane:** It involves adding labels or annotations to inject a sidecar.
- **Data Plane:** It helps in traffic management such as handling NF call flows by adding Service Entries (SE), Destination Rules (DR), Envoy Filters (EF) and other resource changes such as API version changes between versions. Data plane configuration is done manually depending on each NF requirement and ASM deployment.

Data Plane Configuration

The Data Plane configuration consists of the following Custom Resource Definitions (CRDs):

- Service Entry (SE)
- Destination Rule (DR)
- Envoy Filter (EF)
- Peer Authentication (PA)
- Authorization Policy (AP)
- Virtual Service (VS)
- requestAuthentication

Note

Use Helm charts to add or delete CRDs that you may require due to ASM upgrades to configure features across different releases.

The Data Plane configuration is applicable in the following scenarios:

- **NF to NF Communication:** During NF to NF communication where sidecar is injected on both NFs, you need SE and DR to communicate with the other NF, otherwise sidecar rejects the communication. All Egress communications of NFs must have an entry for SE and DR and the same needs to be configured.

Note

For out of cluster communication, you must configure the core DNS with the producer NF endpoint to enable access.

- **Kube-api-server:** For Kube-api-server, a few NF flows may require access to the Kubernetes API server. The ASM proxy (mTLS enabled) may block this. As per the F5 recommendation, the NF requires to add SE for Kubernetes API server in its namespace.
- **Envoy Filters:** When Sidecars rewrite the header value with its default value, the headers from back-end services are lost. To overcome this situation, Envoy Filters help in passing the headers from back-end services to use it as is.

ASM Configuration File

A sample `ocbsf_custom_values_servicemesh_config_26.1.201.yaml` is available in `Custom_Templates` folder. For downloading the file, see [Customizing BSF](#).

2.2.1.10.1 Predeployment configurations

This section explains the pre-deployment configuration procedure to install Cloud Native Core Binding Support Function (BSF) with ASM support.

Step 1 - Creating BSF Namespace

Create a namespace and apply istio injection to it by using the following command:

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

Example

```
kubectl label --overwrite namespace ocbsf istio-injection=enabled
```

Step 2 - The Operator should have special capabilities at service account level to start pre-install init container.

Example of some special capabilities:

```
readOnlyRootFilesystem: false  
allowPrivilegeEscalation: true  
allowedCapabilities:  
- NET_ADMIN  
- NET_RAW  
runAsUser:  
  rule: RunAsAny
```

2.2.1.10.2 Deploying BSF With ASM

Customize the `ocbsf-26.1.201-custom-values-servicemesh-config.yaml` file

To customize the `ocbsf_custom_values_servicemesh_config_26.1.201.yaml` file, uncomment and modify the parameters as per your requirements.

A sample `ocbsf_custom_values_servicemesh_config_26.1.201.yaml` is available in `Custom_Templates` file. For downloading the file, see [Customizing BSF](#).

Note

When BSF is deployed with ASM and cnDBTier is also installed in the same namespace or cluster, then you can skip installing service entries and destination rules.

To update **Service Entries**, make the required changes using the following sample template:

```
#serviceEntries:
# - hosts: |-
#   [ "mysql-connectivity-service.<cnDbTierNamespace>.svc.<clusterName>" ]
#   exportTo: |-
#     [ "." ]
#   location: MESH_EXTERNAL
#   ports:
#     - number: 3306
#       name: mysql
#       protocol: MySQL
#   name: ocbsf-to-mysql-external-se-test
# - hosts: |-
#   [ ".*.<clusterName>" ]
#   exportTo: |-
#     [ "." ]
#   location: MESH_EXTERNAL
#   ports:
#     - number: 8090
#       name: http2-8090
#       protocol: TCP
#     - number: 80
#       name: HTTP2-80
#       protocol: TCP
#   name: ocbsf-to-other-nf-se-test
# - hosts: |-
#   [ "kubernetes.default.svc.<clusterName>" ]
#   exportTo: |-
#     [ "." ]
#   location: MESH_INTERNAL
#   addresses: |-
#     [ "192.168.200.36" ]
#   ports:
#     - number: 443
#       name: https
#       protocol: HTTPS
#   name: nf-to-kube-api-server
```

To customize **Destination Rule**, make the required changes using the following sample template:

```
# destinationRules:
# - host: ".*.<clusterName>"
#   mode: DISABLE
#   name: ocbsf-to-other-nf-dr-test
#   sbitimers: true
```

```
# tcpConnectTimeout: "750ms"
# tcpKeepAliveProbes: 3
# tcpKeepAliveTime: "1500ms"
# tcpKeepAliveInterval: "1s"
# - host: mysql-connectivity-service.<cnDBTiernamespace>.svc.cluster.local
# mode: DISABLE
# name: mysql-occne
# sbitimers: false
```

For customizing **envoyFilters** according to the Istio version installed on the Bastion server, use any of the following templates:

For Istio version 1.11.x and 1.14.x

```
#envoyFilters_v_19x_111x:
# - name: set-xfcc-bsf
#   labelselector: "app.kubernetes.io/instance: ocbsf"
#   applyTo: NETWORK_FILTER
#   filtername: envoy.filters.network.http_connection_manager
#   operation: MERGE
#   typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
#   configuration:
#     - configkey: forward_client_cert_details
#       configvalue: ALWAYS_FORWARD_ONLY
# - name: serverheaderfilter
#   labelselector: "app.kubernetes.io/instance: ocbsf"
#   applyTo: NETWORK_FILTER
#   filtername: envoy.filters.network.http_connection_manager
#   operation: MERGE
#   typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
#   configuration:
#     - configkey: server_header_transformation
#       configvalue: PASS_THROUGH
# - name: custom-http-stream
#   labelselector: "app.kubernetes.io/instance: ocbsf"
#   applyTo: NETWORK_FILTER
#   filtername: envoy.filters.network.http_connection_manager
#   operation: MERGE
#   typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
#   configuration:
#     - configkey: server_header_transformation
#       configvalue: PASS_THROUGH
#   stream_idle_timeout: "6000ms"
#   max_stream_duration: "7000ms"
#   patchContext: SIDECAR_OUTBOUND
#   networkFilter_listener_port: 8000
# - name: custom-tcpsocket-timeout
#   labelselector: "app.kubernetes.io/instance: ocbsf"
#   applyTo: FILTER_CHAIN
```

```
# patchContext: SIDECAR_INBOUND
# operation: MERGE
# transport_socket_connect_timeout: "750ms"
# filterChain_listener_port: 8000
# - name: custom-http-route
# labelselector: "app.kubernetes.io/instance: ocbsf"
# applyTo: HTTP_ROUTE
# patchContext: SIDECAR_OUTBOUND
# operation: MERGE
# route_idle_timeout: "6000ms"
# route_max_stream_duration: "7000ms"
# httpRoute_routeConfiguration_port: 8000
# vhostname: "bsf-ocbsf-policy-ds.ocbsf.svc.cluster:8000"
```

For Istio version 1.11.x and 1.14.x

Note

Istio 1.11.x and 1.14.x support the same template for **envoyFilters** configurations.

```
envoyFilters_v_19x_111x:
- name: xfccfilter
  labelselector: "app.kubernetes.io/instance: ocbsf"
  configpatch:
    - applyTo: NETWORK_FILTER
      filtername: envoy.filters.network.http_connection_manager
      operation: MERGE
      typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionMana
ger
    configuration:
      - configkey: forward_client_cert_details
        configvalue: ALWAYS_FORWARD_ONLY
- name: serverheaderfilter
  labelselector: "app.kubernetes.io/instance: ocbsf"
  configpatch:
    - applyTo: NETWORK_FILTER
      filtername: envoy.filters.network.http_connection_manager
      operation: MERGE
      typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionMana
ger
    configuration:
      - configkey: server_header_transformation
        configvalue: PASS_THROUGH
- name: custom-http-stream
  labelselector: "app.kubernetes.io/instance: ocbsf"
  configpatch:
    - applyTo: NETWORK_FILTER
      filtername: envoy.filters.network.http_connection_manager
      operation: MERGE
      typeconfig: type.googleapis.com/
```

```

envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionMan
ger
    configuration:
      - configkey: server_header_transformation
        configvalue: PASS_THROUGH
        stream_idle_timeout: "6000ms"
        max_stream_duration: "7000ms"
        patchContext: SIDECAR_OUTBOUND
        networkFilter_listener_port: 8000
  - name: custom-tcpsocket-timeout
    labelselector: "app.kubernetes.io/instance: ocbsf"
    configpatch:
      - applyTo: FILTER_CHAIN
        patchContext: SIDECAR_INBOUND
        operation: MERGE
        transport_socket_connect_timeout: "750ms"
        filterChain_listener_port: 8000
  - name: custom-http-route
    labelselector: "app.kubernetes.io/instance: ocbsf"
    configpatch:
      - applyTo: HTTP_ROUTE
        patchContext: SIDECAR_OUTBOUND
        operation: MERGE
        route_idle_timeout: "6000ms"
        route_max_stream_duration: "7000ms"
        httpRoute_routeConfiguration_port: 8000
        vhostname: "ocbsf.svc.cluster:8000"
  - name: logicaldnscluster
    labelselector: "app.kubernetes.io/instance: ocbsf"
    configpatch:
      - applyTo: CLUSTER
        clusterservice: rchltxekvzwcamf-y-ec-
x-002.amf.5gc.mnc480.mcc311.3gppnetwork.org
        operation: MERGE
        logicaldns: LOGICAL_DNS
      - applyTo: CLUSTER
        clusterservice: rchltxekvzwcamd-y-ec-
x-002.amf.5gc.mnc480.mcc311.3gppnetwork.org
        operation: MERGE
        logicaldns: LOGICAL_DNS

```

Note

The parameter *vhostname* is mandatory when *applyTo* is **HTTP_ROUTE**.

Note

Depending on the Istio version, update the correct value of envoy filters in the following line:

```
{{- range .Values.envoyFilters_v_19x_111x }}
```

For customizing **PeerAuthentication**, make the required changes using the following sample template:

```
#peerAuthentication:
# - name: default
#   tlsmode: PERMISSIVE
# - name: cm-service
#   labelselector: "app.kubernetes.io/name: cm-service"
#   tlsmode: PERMISSIVE
# - name: ingress
#   labelselector: "app.kubernetes.io/name: ocbsf-ingress-gateway"
#   tlsmode: PERMISSIVE
# - name: diam-gw
#   labelselector: "app.kubernetes.io/name: diam-gateway"
#   tlsmode: PERMISSIVE
```

Istio **Authorization Policy** enables access control on workloads in the mesh. Authorization policy supports CUSTOM, DENY and ALLOW actions for access control. When CUSTOM, DENY and ALLOW actions are used for a workload at the same time, the CUSTOM action is evaluated first, then the DENY action, and finally the ALLOW action.

For more details on Istio Authorization Policy, see [Istio / Authorization Policy](#).

To customize the Authorization Policy, make the required changes using the following sample template:

```
#authorizationPolicies:
#- name: allow-all-provisioning-on-ingressgateway-ap
#   labelselector: "app.kubernetes.io/name: ingressgateway"
#   action: "ALLOW"
#   hosts:
#     - "*"
#   paths:
#     - "/nudr-dr-prov/*"
#     - "/nudr-dr-mgm/*"
#     - "/nudr-group-id-map-prov/*"
#     - "/slf-group-prov/*"
#- name: allow-all-sbi-on-ingressgateway-ap
#   labelselector: "app.kubernetes.io/name: ingressgateway"
#   action: "ALLOW"
#   hosts:
#     - "*"
#   paths:
#     - "/nbsf-policyauthorization/*"
#   xfccvalues:
#     - "*DNS=nrf1.site1.com"
#     - "*DNS=nrf2.site2.com"
#     - "*DNS=scp1.site1.com"
#     - "*DNS=scp1.site2.com"
#     - "*DNS=scp1.site3.com"
```

VirtualService is required to configure the retry attempts for the destination host. For instance, for error response code value 503, the default behaviour of Istio is to retry two times. However, if the user wants to configure the number of retry attempts, then it can be done using virtualService.

In the following example, the number of retry attempts are set to 0:

```
#virtualService:
# - name: scp1site1vs
#   host: "scp1.site1.com"
#   destinationhost: "scp1.site1.com"
#   port: 8000
#   exportTo: |-
#     [ "." ]
#   attempts: "0"
#   timeout: 7s
# - name: scp1site2vs
#   host: "scp1.site2.com"
#   destinationhost: "scp1.site2.com"
#   port: 8000
#   exportTo: |-
#     [ "." ]
#   retryon: 5xx
#   attempts: "1"
#   timeout: 7s
```

where, host or destination name uses the format - <release_name>-<egress_svc_name>

To get the <egress_svc_name>, run the following command:

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

Note

For 5xx response codes, set the value of retry attempts to 1.

Request Authentication is used to configure JWT tokens for OAuth validation. Network functions need to authenticate the OAuth token sent by consumer network functions by using the Public key of the NRF signing certificate and using service mesh to authenticate the token. Using the following sample format, users can configure **requestAuthentication** as per their system requirements:

```
requestAuthentication:
# - name: jwttokenwithjson
#   labelselector: httpbin
#   issuer: "jwtissue"
#   jwks: |-
#     '{
#       "keys": [{
#         "kid": "1",
#         "kty": "EC",
#         "crv": "P-256",
#         "x": "Qr15t1-Apuj8uRI2o_BP9loqvaBnyM40TPAD_peDe4",
#         "y": "Y7vNMKGNAtlteMV-KJIaG-0UlcVRGFhtUVI8ZoXIzRY"
#       }]
#     }'
```

```
# issuer: "jwtissue"
# jwksUri: https://example.com/.well-known/jwks.json
```

① Note

For requestAuthetication, use either jwks or jwksUri.

Run the following command to create the Custom Resource Definitions (CRDs):

```
helm install ocbsf-servicemesh-config ocbsf-servicemesh-config-26.1.201.tgz -
n ocbsf -f ocbsf_custom_values_servicemesh_config_26.1.201.yaml
```

2.2.1.10.3 Post-deployment Configurations

To verify if the CDRs have been created, run the following command::

```
kubectl get
se,dr,peerauthentication,envoyfilter,vs,authorizationpolicy,requestauthenticat
ion -n ocbsf
```

The following is a sample output:

```
NAME
HOSTS                                LOCATION
RESOLUTION    AGE
serviceentry.networking.istio.io/nf-to-kube-api-server
["kubernetes.default.svc.vega"]      MESH_INTERNAL
NONE          17h
serviceentry.networking.istio.io/vega-ns1a-to-mysql-external-se-test
["mysql-connectivity-service.vega-ns1.svc.vega"]  MESH_EXTERNAL
NONE          17h
serviceentry.networking.istio.io/vega-ns1a-to-other-nf-se-test
["*.vega"]                          MESH_EXTERNAL
NONE
17hNAME
HOST                                AGE
destinationrule.networking.istio.io/jaeger-dr      occne-
tracer-jaeger-query.occne-infra                    17h
destinationrule.networking.istio.io/mysql-occne    mysql-
connectivity-service.vega-ns1.svc.cluster.local    17h
destinationrule.networking.istio.io/prometheus-dr  occne-
prometheus-server.occne-infra                      17h
destinationrule.networking.istio.io/vega-ns1a-to-other-nf-dr-test
*.vega
17hNAME                                MODE    AGE
peerauthentication.security.istio.io/cm-service    PERMISSIVE  17h
peerauthentication.security.istio.io/default        PERMISSIVE  17h
peerauthentication.security.istio.io/diam-gw        PERMISSIVE  17h
peerauthentication.security.istio.io/ingress         PERMISSIVE  17h
peerauthentication.security.istio.io/ocats-policy    PERMISSIVE
17hNAME                                AGE
envoyfilter.networking.istio.io/ocats-policy-xfcc    17h
```

```

envoyfilter.networking.istio.io/serverheaderfilter      17h
envoyfilter.networking.istio.io/serverheaderfilter-nf1stub 17h
envoyfilter.networking.istio.io/serverheaderfilter-nf2stub 17h
envoyfilter.networking.istio.io/set-xfcc-bsf
17hNAME                                                  GATEWAYS
HOSTS                                                    AGE
virtualservice.networking.istio.io/nrfvirtual1          ["vega-ns1a-occp-
egress-gateway"] 17h
[cloud-user@vega-bastion-1 ~]$

```

Then, perform the steps described in [Installing BSF Package](#).

2.2.1.10.4 Deleting Service Mesh

This section describes the steps to delete Aspen Service Mesh (ASM) for ASM based BSF.

1. Disable ASM.

```
kubectl label --overwrite <namespace> ocbsf istio-injection=disabled
```

where,

namespace-name is the deployment namespace used by helm command.

2. Delete all the pods in the namespace.

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

3. Delete ASM.

```
helm delete <helm-release-name> -n <namespace-name>
```

where, helm-release-name is the release name used by the helm install command. This release name must be the same as the release name used for ServiceMesh.

namespace-name is the deployment namespace used by helm command.

Example:

```
helm delete ocbsf-servicemesh-config -n ocbsf
```

4. Verify ASM deletion.

```
kubectl get se,dr,peerauthentication,envoyfilter,vs -n ocbsf
```

2.2.1.11 Configuring Network Policies

Network Policies allow you to define ingress or egress rules based on Kubernetes resources such as Pod, Namespace, IP, and Port. These rules are selected based on Kubernetes labels in the application. These Network Policies enforce access restrictions for all the applicable data flows except communication from Kubernetes node to pod for invoking container probe.

Note

Configuring Network Policy is optional. Based on the security requirements, Network Policy can be configured.

For more information on Network Policies, see <https://kubernetes.io/docs/concepts/services-networking/network-policies/>.

Note

- If the traffic is blocked or unblocked between the pods even after applying Network Policies, check if any existing policy is impacting the same pod or set of pods that might alter the overall cumulative behavior.
- If changing default ports of services such as Prometheus, Database, Jaegar, or if Ingress or Egress Gateway names is overridden, update them in the corresponding Network Policies.

Configuring Network Policies

Network Policies support Container Network Interface (CNI) plugins for cluster networking.

Note

For any deployment with CNI, it must be ensured that Network Policy is supported.

Following are the various operations that can be performed for Network Policies:

2.2.1.11.1 Installing Network Policies

Prerequisite

Network Policies are implemented by using the network plug-in. To use Network Policies, you must be using a networking solution that supports Network Policy.

Note

For a fresh installation, it is recommended to install Network Policies before installing BSF. However, if BSF is already installed, you can still install the Network Policies.

To install Network Policies:

1. Open the `ocbsf-network-policy-custom-values.yaml` file provided in the release package zip file.
For downloading the file, see [Downloading BSF Package](#) and [Pushing the Images to Customer Docker Registry](#).
2. The file is provided with the default Network Policies. If required, update the `ocbsf-network-policy-custom-values.yaml` file. For more information on the parameters, see [Configuration Parameters for Network Policies](#).

Note

- To run ATS, uncomment the following policies from `ocbsf-network-policy-custom-values.yaml`:
 - `allow-egress-for-ats`
 - `allow-ingress-to-ats`
 - `allow-egress-to-ats-pods-from-bsf-pods`
 - `allow-ingress-from-ats-pods-to-bsf-pods`
- To connect with CNC Console, update the following parameter in the `allow-ingress-from-console` Network Policy in the `ocbsf-network-policy-custom-values.yaml`:

```
kubernetes.io/metadata.name: <namespace in which CNCC is
deployed>
```

- In `allow-ingress-prometheus` policy, `kubernetes.io/metadata.name` parameter must contain the value for the namespace where Prometheus is deployed, and `app.kubernetes.io/name` parameter value should match the label from Prometheus pod.

3. Run the following command to install the Network Policies:

```
helm install <helm-release-name> ocbsf-network-policy/ -n <namespace> -f
<custom-value-file>
```

where:

- `<helm-release-name>` is the `ocbsf-network-policy` Helm release name.
- `<yaml-file>` is the `ocbsf-network-policy-custom-value` file.
- `<namespace>` is the OCBSF namespace.

For example:

```
helm install ocbsf-network-policy ocbsf-network-policy/ -n ocbsf
-f ocbsf-network-policy-custom-values.yaml
```

Note

- Connections that were created before installing Network Policy and still persist are not impacted by the new Network Policy. Only the new connections would be impacted.
- If you are using ATS suite along with Network Policies, it is required to install the BSF and ATS in the same namespace.
- It is highly recommended to run ATS after deploying Network Policies to detect any missing/invalid rule that can impact signaling flows.

2.2.1.11.2 Upgrading Network Policies

To add, delete, or update Network Policies:

1. Modify the `ocbsf-network-policy-custom-values.yaml` file to update, add, or delete the Network Policy.
2. Run the following command to upgrade the Network Policies:

```
helm upgrade <helm-release-name> network-policy/ -n <namespace> -f  
  <custom-value-file>
```

where:

- `<helm-release-name>` is the `ocbsf-network-policy` helm release name.
- `<yaml-file>` is the `ocbsf-network-policy-custom-value` file.
- `<namespace>` is the OCBSF namespace.

For example:

```
helm upgrade ocbsf-network-policy ocbsf-network-policy/ -n ocbsf  
  -f ocbsf-network-policy-custom-values.yaml
```

2.2.1.11.3 Verifying Network Policies

Run the following command to verify if the Network Policies are deployed successfully:

```
kubectl get <helm-release-name> -n <namespace>
```

For example:

```
kubectl get ocbsf-network-policy -n ocbsf
```

Where,

- `helm-release-name`: `ocbsf-network-policy` Helm release name.
- `namespace`: CNC Console namespace.

2.2.1.11.4 Uninstalling Network Policies

Run the following command to uninstall network policies:

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

For example:

```
helm uninstall ocbsf-network-policy -n ocbsf
```

Note

While using the debug container, it is recommended to uninstall the network policies or update them as required to establish the connections.

2.2.1.11.5 Configuration Parameters for Network Policies

Table 2-10 Supported Kubernetes Resource for Configuring Network Policies

Parameter	Description	Details
apiVersion	This is a mandatory parameter. Specifies the Kubernetes version for access control. Note: This is the supported api version for network policy. This is a read-only parameter.	Data Type: string Default Value: networking.k8s.io/v1
kind	This is a mandatory parameter. Represents the REST resource this object represents. Note: This is a read-only parameter.	Data Type: string Default Value: NetworkPolicy

Table 2-11 Supported Parameters for Configuring Network Policies

Parameter	Description	Details
metadata.name	This is a mandatory parameter. Specifies a unique name for Network Policies.	Data Type: String Default Value: {{ .metadata.name }}
spec.{}	This is a mandatory parameter. This consists of all the information needed to define a particular network policy in the given namespace. Note: Policy supports the spec parameters defined in "Supported Kubernetes Resource for Configuring Network Policies".	Default Value: NA

For more information, see *Network Policies* in *Oracle Communications Cloud Native Core, Binding Support Function User Guide*.

2.2.1.12 Configuring Traffic Segregation

This section provides information on how to configure Traffic Segregation in BSF. For description of "Traffic Segregation" feature, see "Traffic Segregation" section in "BSF Features" chapter of *Oracle Communications Cloud Native Core, Binding Support Function User Guide*.

Various networks can be created at the time of CNE cluster installation. The following things can be customized at the time of the cluster installation using `cnlb.ini` file provided as part of CNE installation.

1. Number of network pools
2. Number of Egress IPs
3. Number of Service IPs/Ingress IPs
4. External IPs/subnet

For more information, see *Oracle Communications Cloud Native Core, Cloud Native Environment User Guide*.

Note

- The network attachments will be deployed as a part of cluster installation only.
- The network attachment name should be unique for all the pods.
- The destination (egress) subnet addresses are known beforehand and defined under `cnlb.ini` file `egress_dest` variable to generate Network Attachment Definitions.

Note

When BSF is deployed with `cnLB` feature enabled, the `TYPE` field for the applicable BSF services shall remain to be "LoadBalancer" and the `EXTERNAL-IP` field shall be in pending state. This has no impact on the overall `cnLB` functionality in BSF application.

Configuration at Ingress Gateway

To use one or multiple interfaces, you must configure annotations in the `ingress-gateway.deployment.customExtension.annotations` parameter of the `ocbsf_custom_values_26.1.201.yaml` file.

```
ingress-gateway:
  deployment:
    customExtension:
      annotations: {
        # Enable this section for service-mesh based installation
        #   traffic.sidecar.istio.io/excludeOutboundPorts: "9000,8095,8096",
        #   traffic.sidecar.istio.io/excludeInboundPorts: "9000,8095,8096"
      }
```

Annotation for a single interface

```
k8s.v1.cni.cncf.io/networks: default/<network interface>@<network interface>,
oracle.com.cnlb: '[{"backendPortName": "<igw port name>", "cnlbIp":
"<external IP>","cnlbPort":"<port number>"}]'
```

Here,

- `k8s.v1.cni.cncf.io/networks`: Contains all the network attachment information the pod uses for network segregation.
- `oracle.com.cnc/cnlb`: To define service IP and port configurations that the deployment will employ for ingress load balancing.

Where,

- **cnlbIp** is the front-end IP utilized by the application.
- **cnlbPort** is the front-end port used in conjunction with the CNLB IP for load balancing.
- **backendPortName** is the backend port name of the container that needs load balancing, retrievable from the deployment or pod spec of the application.

Note

In case of TLS enabled for Ingress gateway, please use `backendPortName` as **igw-https**.

Sample annotation for a single interface:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig1-int8@nf-sig1-int8,
oracle.com.cnc/cnlb: '[{"backendPortName": "igw-http", "cnlbIp":
"10.123.155.16", "cnlbPort": "80"}]'
```

Annotation for two or multiple interfaces

```
k8s.v1.cni.cncf.io/networks: default/<network interface1>@<network
interface1>, default/<network interface2>@<network interface2>,
oracle.com.cnc/cnlb: '[{"backendPortName": "<igw port name>", "cnlbIp":
"<network interface1>/<external IP1>, <network interface2>/<external
IP2>", "cnlbPort": "<port number>"}]',
oracle.com.cnc/ingressMultiNetwork: "true"
```

Sample annotation for two or multiple interfaces:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig1-int8@nf-sig1-int8,default/nf-
sig2-int9@nf-sig2-int9,
oracle.com.cnc/cnlb: '[{"backendPortName": "igw-http", "cnlbIp": "nf-sig1-
int8/10.123.155.16,nf-sig2-int9/10.123.155.30", "cnlbPort": "80"}]',
oracle.com.cnc/ingressMultiNetwork: "true"
```

Sample annotation for multiport:

```
k8s.v1.cni.cncf.io/networks: default/nf-oam-int5@nf-oam-int5,
oracle.com.cnc/cnlb: '[{"backendPortName": "query", "cnlbIp":
"10.75.180.128", "cnlbPort": "80"},
{"backendPortName": "admin", "cnlbIp": "10.75.180.128", "cnlbPort": "16687"}]'
```

In the above example, each item in the list refers to a different backend port name with the same CNLB IP, but the ports for the front end are distinct.

Ensure that the backend port name aligns with the container port name specified in the deployment's specification, which needs to be load balanced from the port list. The CNLB IP represents the external IP of the service, and cnlbPort is the external-facing port:

```
ports:
- containerPort: 16686
  name: query
  protocol: TCP
- containerPort: 16687
  name: admin
  protocol: TCP
```

Configuration at Egress Gateway

To use one or multiple interfaces, you must configure annotations in the egress-gateway.deployment.customExtension.annotations parameter of the ocsf_custom_values_26.1.201.yaml file.

```
egress-gateway:
  deployment:
    customExtension:
      annotations: {
        # Enable this section for service-mesh based installation
        # traffic.sidecar.istio.io/excludeOutboundPorts: "9000,8095,8096",
        # traffic.sidecar.istio.io/excludeInboundPorts: "9000,8095,8096"
      }
```

Sample annotation for a single interface:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig-egr1@nf-sig-egr1
```

Sample annotation for a multiple interface:

```
k8s.v1.cni.cncf.io/networks: default/nf-oam-egr1@nf-oam-egr1,default/nf-sig-egr1@nf-sig-egr1
```

Configuration at Diameter Gateway

Diameter gateway uses ingress-egress type of NAD to enable traffic flow for both ingress and egress directions.

To use one or multiple interfaces, you must configure annotations in the diameter-gateway.deployment.customExtension.annotations parameter of the ocsf_custom_values_26.1.201.yaml file.

```
diameter-gateway:
  deployment:
    customExtension:
      annotations: {
```

```
# Enable this section for service-mesh based installation
# traffic.sidecar.istio.io/excludeOutboundPorts: "9000,5801",
# traffic.sidecar.istio.io/excludeInboundPorts: "9000,5801"
}
```

Annotation for a single interface:

```
k8s.v1.cni.cncf.io/networks: default/<network interface>@<network interface>,
oracle.com.cnc/cnlb: '[{"backendPortName":"diam-signaling", "cnlbIp":
"<externalIP>","cnlbPort":"<port number>"}]'
```

Here,

- **k8s.v1.cni.cncf.io/networks:** Contains all the network attachment information the pod uses for network segregation.
- **oracle.com.cnc/cnlb:** To define service IP and port configurations that the deployment will employ for diameter gateway ingress load balancing.
- **cnlbIp:** This is the front-end IP utilized by the application
- **cnlbPort:** This is the front-end port used in conjunction with the CNLB IP for load balancing.
- **backendPortName:** This is the backend port name of the container that needs load balancing, retrievable from the deployment or pod spec of the application.

Note

In case of TLS enabled for diameter gateway, please use backendPortName as **tls-signaling**.

Sample annotation for a single interface:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig1-ie1@nf-sig1-ie1,
oracle.com.cnc/cnlb: '[{"backendPortName":"diam-
signaling","cnlbIp":"10.123.155.17","cnlbPort":"3868"}]'
```

Annotation for two or multiple interfaces:

```
k8s.v1.cni.cncf.io/networks: default/<network interface1>@<network
interface1>, default/<networkinterface2>@<network interface2>,
oracle.com.cnc/cnlb: '[{"backendPortName":"diam-signaling","cnlbIp":
"<networkinterface1>/<external IP1>,<networkinterface2>/
<externalIP2>","cnlbPort":"<portnumber>"}] ',
oracle.com.cnc/ingressMultiNetwork: "true"
```

Sample annotation for two or multiple interfaces:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig3-ie1@nf-sig3-ie1,default/nf-sig4-
ie1@nf-sig4-ie1,
oracle.com.cnc/cnlb: '[{"backendPortName":"diam-signaling","cnlbIp":"nf-sig3-
```

```
ie1/10.123.155.16,nf-sig4-ie1/10.123.155.30","cnlbPort":"3868"}]'],  
oracle.com.cnc/ingressMultiNetwork: "true"
```

Sample annotation for multiport:

```
k8s.v1.cni.cncf.io/networks: default/nf-sig3-ie1@nf-sig3-ie1,  
oracle.com.cnc/cnlb:'[{"backendPortName": "query", "cnlbIp":  
"10.75.180.128","cnlbPort": "3868"},  
{"backendPortName": "admin", "cnlbIp": "10.75.180.128","cnlbPort":"16687"}]'
```

In the above sample, each item in the list refers to a different backend port name with the same CNLB IP, but the ports for the front end are distinct.

Ensure that the backend port name aligns with the container port name specified in the deployment's specification, which needs to be load balanced from the port list. The CNLB IP represents the external IP of the service, and cnlbPort is the external-facing port:

```
ports:  
-containerPort: 16686  
  name: query  
  protocol: TCP  
-containerPort: 16687  
  name: admin  
  protocol: TCP
```

For information about the above mentioned annotations, see "Configuring Cloud Native Load Balancer (CNLB)" in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

2.2.2 Installation Tasks

This section explains how to install BSF.

① Note

- Before installing BSF, you must complete [Prerequisites](#) and [Preinstallation Tasks](#).
- In a georedundant deployment, perform the steps explained in this section on all the georedundant sites.

2.2.2.1 Downloading BSF Package

This section provides information on how to download Binding Support Function (BSF) package.

To download the BSF package from MOS, perform the following steps:

1. Log in to [My Oracle Support](#) with your credentials.
2. Select **Patches and Updates** tab to locate the patch.
3. In **Patch Search** window, click **Product or Family (Advanced)** tab.

4. Enter Oracle Communications Cloud Native Core - 5G in the **Product** field.
5. From the release drop-down list, select Oracle Communications Cloud Native Core Binding Support Function <Release Number>. Where, <release_number> indicates the required release number of CNC BSF.
6. Click **Search**.
Step Result: The **Patch Advanced Search Results** displays a list of available releases.
7. Select the required patch from the search results.
Step Result: The Patch Details window opens.
8. Click **Download**.
Step Result: File Download window opens.
9. Click the <p*****_<release_number>_Tekelec>.zip file to download the BSF release package.

2.2.2.2 Pushing the Images to Customer Docker Registry

BSF deployment package includes ready-to-use images and Helm charts to orchestrate containers in Kubernetes.

Table 2-12 Docker Images for BSF

Service Name	Docker Image Name	Image Tag
Alternate Route Service	alternate_route	26.1.206
BSF Management	oc-bsf-management	26.1.201
Application Info Service	oc-app-info	26.1.205
Common Configuration Hook	common_config_hook	26.1.206
CNC Common Configuration Hook	cnc_common_config_hook	26.1.204
Config Server	oc-config-server	26.1.205
Configuration Management Server	oc-config-mgmt	26.1.205
Debug Tool	ocdebug-tools	26.1.202
Diameter Gateway	oc-diam-gateway	26.1.205
Egress Gateway	ocegress_gateway	26.1.206
NF Test	nf_test	26.1.201
Ingress Gateway	ocingress_gateway	26.1.206
Ingress Gateway/Egress Gateway init configuration	configurationinit	26.1.206
Ingress Gateway/Egress Gateway update configuration	configurationupdate	26.1.206
NRF Client Service	nrf-client	26.1.204
Performance Monitoring Service	oc-perf-info	26.1.205
Query Service	oc-query	26.1.205
Session State Audit	oc-audit	26.1.205

Pushing images

To push the images to the registry:

1. Run the following command to untar the BSF package file to get the BSF docker image tar file:

```
tar -xvzf <ReleaseName>-pkg-<Releasenum>.tgz
```

Example: `tar -xvzf ocbsf-pkg-26.1.201.0.0.tgz`

The directory consists of the following:

- **BSF Docker Images File:**
ocbsf-images-26.1.201.tar
- **Helm File:**
ocbsf-26.1.201.tgz
- **Readme txt File:**
Readme.txt
- **Checksum for Helm chart tgz file:**
ocbsf-26.1.201.tgz.sha256
- **Checksum for Helm chart for Service Mesh tgz file:**
ocbsf-servicemesh-config-26.1.201.tgz.sha256
- **Checksum for images' tgz file:**
ocbsf-images-26.1.201.tar.sha256

2. Run one of the following commands to load the ocbsf-images-<release_number>.tar file:

```
docker load --input /IMAGE_PATH/ocbsf-images-26.1.201.tar
```

where *IMAGE_PATH* points to the location where ocbsf-images-26.1.201.tar is stored.

For CNE 1.8.0 and later versions, use the following command:

```
podman load --input /IMAGE_PATH/ocbsf-images-26.1.201.tar
```

3. Run one of the following commands to verify that the images are loaded:

```
docker images
```

```
podman images
```

Verify the list of images shown in the output with the list of images shown in the table [Table 2-12](#). If the list does not match, reload the image tar file.

For more information on docker images available in BSF, see [Docker Images](#).

4. Run one of the following commands to tag the images to the registry:

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

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

5. Run one of the following commands to push the images to the registry:

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

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

Note

It is recommended to configure the Docker certificate before running the push command to access customer registry through HTTPS, otherwise docker push command may fail.

Example for OCCNE 1.8.0 and later versions

```
podman tag docker.io/ocbsf/oc-config-mgmt:26.1.205 occne-repo-host/oc-config-  
mgmt:26.1.205  
podman push occne-repo-host/oc-config-mgmt:26.1.205
```

```
podman tag docker.io/ocbsf/nf_test:26.1.201 occne-repo-host/nf_test:26.1.201  
podman push occne-repo-host/nf_test:26.1.201
```

```
podman tag docker.io/ocbsf/alternate_route:26.1.206 occne-repo-host/  
alternate_route:26.1.206  
podman push occne-repo-host/alternate_route:26.1.206
```

```
podman tag docker.io/ocbsf/oc-config-server:26.1.205 occne-repo-host/oc-  
config-server:26.1.205  
podman push occne-repo-host/oc-config-server:26.1.205
```

```
podman tag docker.io/ocbsf/configurationupdate:26.1.206 occne-repo-host/  
configurationupdate:26.1.206  
podman push occne-repo-host/configurationupdate:26.1.206
```

```
podman tag docker.io/ocbsf/oc-app-info:26.1.205 occne-repo-host/oc-app-  
info:26.1.205  
podman push occne-repo-host/oc-app-info:26.1.205
```

```
podman tag docker.io/ocbsf/ocingress_gateway:26.1.206 occne-repo-host/  
ocingress_gateway:26.1.206  
podman push occne-repo-host/ocingress_gateway:26.1.206
```

```
podman tag docker.io/ocbsf/ocegress_gateway:26.1.206 occne-repo-host/  
ocegress_gateway:26.1.206  
podman push occne-repo-host/ocegress_gateway:26.1.206
```

```
podman tag docker.io/ocbsf/oc-diam-gateway:26.1.205 occne-repo-host/oc-diam-  
gateway:26.1.205  
podman push occne-repo-host/oc-diam-gateway:26.1.205
```

```
podman tag docker.io/ocbsf/oc-bsf-management:26.1.201 occne-repo-host/oc-bsf-  
management:26.1.201  
podman push occne-repo-host/oc-bsf-management:26.1.201
```

```
podman tag docker.io/ocbsf/oc-query:26.1.205 occne-repo-host/oc-query:26.1.205  
podman push occne-repo-host/oc-query:26.1.205
```

```
podman tag docker.io/ocbsf/oc-perf-info:26.1.205 occne-repo-host/oc-perf-  
info:26.1.205  
podman push occne-repo-host/oc-perf-info:26.1.205
```

```
podman tag docker.io/ocbsf/nrf-client:26.1.204 occne-repo-host/nrf-
```

```
client:26.1.204
podman push occne-repo-host/nrf-client:26.1.204

podman tag docker.io/ocbsf/configurationinit:26.1.206 occne-repo-host/
configurationinit:26.1.206
podman push occne-repo-host/configurationinit:26.1.206

podman tag docker.io/ocbsf/cnc_common_config_hook:26.1.204 occne-repo-host/
cnc_common_config_hook:26.1.204
podman push occne-repo-host/cnc_common_config_hook:26.1.204

podman tag docker.io/ocbsf/common_config_hook:26.1.206 occne-repo-host/
common_config_hook:26.1.206
podman push occne-repo-host/common_config_hook:26.1.206

podman tag docker.io/ocbsf/ocdebug-tools:26.1.202 occne-repo-host/ocdebug-
tools:26.1.202
podman push occne-repo-host/ocdebug-tools:26.1.202

podman tag docker.io/ocbsf/oc-audit:26.1.205 occne-repo-host/oc-audit:26.1.205
podman push occne-repo-host/oc-audit:26.1.205
```

2.2.2.3 Installing BSF Package

This section describes how to install BSF package.

To install BSF package, perform the following steps:

1. Unzip the release package to the location where you want to install BSF. The package is as follows `ReleaseName_pkg_Releasenumver.tgz`:
where:

ReleaseName is a name that is used to track this installation instance.

Releasenumver is the release number. For example, `ocbsf_pkg_26.1.201_0_0.tgz`.

Run the following command to access the extracted package:
2. Customize the `ocbsf_custom_values_26.1.201.yaml` file. For more information, see [Customizing BSF](#).

Note

The values of the parameters mentioned in the `ocbsf_custom_values_26.1.201.yaml` file overrides the default values specified in the Helm chart. If the `envMySQLDatabase` parameter is modified, you must modify the `configDbName` parameter with the same value.

Note

The URL syntax for perf-info must be in the correct syntax otherwise, it keeps restarting. The following is a URL example for the bastion server if the BSF is deployed on OCCNE platform. On any other PaaS platform, the url should be updated according to the *Prometheus* and *Jaeger* query deployment.

```
# Values provided must match the Kubernetes environment.
perf-info:
  configmapPerformance:
    prometheus: http://occne-prometheus-server.occne-infra.svc/
    clustername/prometheus
    jaeger:jaeger-agent.occne-infra
    jaeger_query_url:http://jaeger-query.occne-infra/clustername/
  jaeger
```

At least three configuration items must be present in the config map for perf-info, failing which perf-info will not work. If jaeger is not enabled, the jaeger and jaeger_query_url parameter can be omitted.

3. Install BSF using Helm:

```
helm install <release-name> -f <custom_file> <helm-chart> -namespace
<release-namespace> --atomic --timeout 10m
```

where:

- helm-chart is the location of the Helm chart extracted from ocbsf-pkg-26.1.201.0.0.tgz file
- release_name is the release name used by Helm command. The maximum allowed length is 63 characters.
- release_namespace is the deployment namespace used by helm command.
- custom_file is the name of the custom values yaml file (including location).

Note

To verify installation while running the install command, run the following command on a separate window:

```
watch kubectl get jobs,pods -n release_namespace
```

For example:

```
helm install ocbsf /home/cloud-user/bsf-26.1.201.0.0.tgz --namespace ocbsf
-f
  ocbsf-custom-values-26.1.201.yaml --atomic
```

Parameters in helm install command:

- **atomic**: If this parameter is set, the installation process purges the Helm chart on failure. The wait flag will be set automatically.
- **wait**: If this parameter is set, installation process will wait until all pods, PVCs, Services, and minimum number of pods of a deployment, StatefulSet, or ReplicaSet are in a ready state before marking the release as successful. It will wait for as long as --timeout.
- **timeout duration** (optional): This parameter specifies the wait time for individual Kubernetes operations such as Jobs for hooks. The default value is 300s (in seconds) in Helm. If the helm install command fails to create a Kubernetes object, it will internally call the purge to delete after reaching the default timeout value.

Note

Timeout value is not for the overall install but for automatic purge on installation failure.

Caution

When you run the install command, make sure that you do not exit from the helm install command manually. After running the helm install command, installing all the services may take some time. In the meantime, you must not press "ctrl+c" to come out from the helm install command as it leads to anomalous behavior.

Note

In Georedundant deployment, if you want to add or remove a site, refer to <Appendix B and Appendix C>.

Note

The following warnings must be ignored for BSF installation on CNE 25.1.2xx and 25.2.1xx :

```
helm install <release-name> -f <custom.yaml> <tgz-file> -n
<namespace>
W0301 07:39:30.096125 1718193 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port
definition with spec.template.spec.containers[0].ports[1]
W0301 07:39:34.033420 1718193 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port
definition with spec.template.spec.containers[0].ports[2]
NAME: <release-name>
LAST DEPLOYED: <Date-Time>
NAMESPACE: <namespace>
STATUS: deployed
REVISION: <N>
```

4. Press "Ctrl+C" to exit watch mode. Make sure to run the watch command on a different terminal.

For Helm 2:

```
helm status <helm-release> -n <namespace>
```

2.2.3 Postinstallation Tasks

This section explains the postinstallation tasks for BSF.

2.2.3.1 Verifying BSF Installation

To verify if BSF is installed:

1. Run the following command to verify the installation status:

For Helm:

```
helm status <helm-release> -n <namespace>
```

For example:

```
helm status ocbsf -n ocbsf
```

Status should be DEPLOYED.

2. Run the following command to verify if the pods are up and active:

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

For example:

```
kubectl get pod -n ocbsf
```

You should see the status as Running and Ready for all the pods.

3. Run the following command to verify if the services are deployed and active:

```
kubectl get services -n <release_namespace>
```

For example:

```
kubectl get services -n ocbsf
```

① Note

If the installation is not successful or you do not see the status as Running for all the pods, perform the troubleshooting steps mentioned in *Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*.

2.2.3.2 Performing Helm Test

This section describes how to perform sanity check for BSF installation through Helm test. The pods to be checked should be based on the namespace and label selector configured for the Helm test configurations.

Helm Test is a feature that validates successful installation of BSF and determines if the NF is ready to take traffic. The pods that are checked are based on the namespace and label selector configured for the Helm test configurations.

① Note

- Helm test can be performed only on Helm3.
- If `nrf-client-nfmanagement.enablePDBSupport` is set to true in the `custom-values.yaml`, Helm test fails. It is an expected behavior as the mode is active and on standby, the leader pod (`nrf-client-management`) will be in ready state but the follower will not be in ready state, which will lead to failure in the Helm test.

Before running Helm test, complete the Helm test configurations.

To perform the helm test, run the following command:

```
helm test <helm-release_name> -n <namespace>
```

where:

`helm-release-name` is the release name.

`namespace` is the deployment namespace where BSF is installed.

Example:

```
helm test ocbsf -n ocbsf
```

Sample output:

```
Pod ocbsf-helm-test-test pending
Pod ocbsf-helm-test-test pending
Pod ocbsf-helm-test-test pending
Pod ocbsf-helm-test-test running
Pod ocbsf-helm-test-test succeeded
NAME: ocbsf-helm-test
LAST DEPLOYED: Thu May 19 12:22:20 2022
NAMESPACE: ocbsf-helm-test
STATUS: deployed
REVISION: 1
TEST SUITE:      ocbsf-helm-test-test
Last Started:    Thu May 19 12:24:23 2022
Last Completed:  Thu May 19 12:24:35 2022
Phase:          Succeeded
```

If the Helm test failed, run the following command to view the logs:

```
helm test <release_name> -n <namespace> --logs
```

Note

- Helm Test expects all of the pods of given microservice to be in READY state for a successful result. However, the NRF Client Management microservice comes with Active/Standby model for the multi-pod support in the current release. When the multi-pod support for NRF Client Management service is enabled, you may ignore if the Helm Test for NRF-Client-Management pod fails.
- If Helm test fails, for details on troubleshooting the installation, see *Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*.

2.2.3.3 Backing Up Important Files

Take a backup of the following files that are required during fault recovery:

- updated ocbf_custom_values_26.1.201.yaml file
- updated ocbf_custom_values_servicemesh_config_26.1.201.yaml file
- updated Helm charts
- secrets, certificates, and keys used during the installation

2.2.3.4 Configuring BSF Alerts

This section describes how to configure alerts for Oracle Communications Cloud Native Core, Binding Support Function. The Alert Manager uses the Prometheus measurements values as reported by microservices in conditions under alert rules to trigger alerts.

Note

- The Alertmanager and Prometheus tools must run in CNE namespace, for example, occne-infra.
- Alert file is packaged with BSF Custom Templates. The **BSF Custom Templates.zip** file can be downloaded from MOS. Unzip the **BSF Custom Templates.zip** file to get **BSF_Alertrules.yaml** file. This file must be readily available before the user configures alerts in Prometheus.

Configuring Alerts for CNE versions prior to 1.5

To Configure BSF alerts in Prometheus:

1. Run the following command to find the configmap and configure alerts in the Prometheus server:

```
kubectl get configmap -n <Namespace>
```

Where:

<Namespace> is the prometheus server namespace used in Helm install command.

For Example, assuming Prometheus server is under **occne-infra** namespace, run the following command to find the configmap:

```
kubectl get configmaps -n occne-infra | grep Prometheus-server
```

Output: occne-prometheus-server 4 46d

2. Run the following command to take a backup of the current Prometheus server configmap:

```
kubectl get configmaps <Name> -o yaml -n <Namespace> > /tmp/  
t_mapConfig.yaml
```

where, <Name> is the Prometheus configmap name used in Helm install command.

3. Check if **alertsbsf** is present in the **t_mapConfig.yaml** file by running the following command:

```
cat /tmp/t_mapConfig.yaml | grep alertsbsf
```

Depending on the outcome of the previous step, perform any of the following:

- If **alertsbsf** is present, delete the **alertsbsf** entry from the **t_mapConfig.yaml** file, by running the following command:

```
sed -i '/etc\/config\/alertsbsf/d' /tmp/t_mapConfig.yaml
```

Note

Run this command only once.

- If **alertsbsf** is not present, add the **alertsbsf** entry in the **t_mapConfig.yaml** file by running the following command:

```
sed -i '/rule_files:/a\    \- /etc/config/alertsbsf' /tmp/  
t_mapConfig.yaml
```

Note

Run this command only once.

4. Run the following command to reload the configmap with the modified file:

```
kubectl replace configmap <Name> -f /tmp/t_mapConfig.yaml
```

Note

It is not required for AlertRules.

5. Add **BSF_Alertrules.yaml** file into Prometheus server configmap by running the following command :

```
kubectl patch configmap <Name> -n <Namespace> --type merge --patch
"$(cat <PATH>/BSF_Alertrules.yaml)"
```

where, <PATH> is the location of the **BSF_Alertrules.yaml** file.

6. Restart prometheus-server pod.
7. Verify the alerts in Prometheus GUI.

The following image shows the BSF Alerts:

/etc/config/alerts/BSF_Alertrules.yaml > BSF_ALERTS

BSF_SERVICES_DOWN (2 active)

IngressCreateErrorRateAbove1Percent (2 active)

IngressTotalErrorRateAbove10Percent (1 active)

PCFBindingErrorRateAbove1Percent (3 active)

BSFTrafficRateAboveThreshold (0 active)

IngressDeleteErrorRateAbove1Percent (0 active)

Configuring Alerts for CNE version from 1.5.0 up to 1.8.x

To Configure BSF alerts in Prometheus:

1. Copy the **BSF_Alertrules.yaml** file to the Bastion Host. Place this file in the /var/occne/cluster/<cluster-name>/artifacts/alerts directory on the OCCNE Bastion Host.

```
$ pwd /var/occne/cluster/stark/artifacts/alerts
$ ls
occne_alerts.yaml
$ vi BSF_Alertrules.yaml
$ ls BSF_Alertrules.yaml occne_alerts.yaml
```

2. To set the correct file permissions, run the following command:

```
$ chmod 644 BSF_Alertrules.yaml
```

3. To load the updated rules from the Bastion host in the file to the existing occne-prometheus-alerts configmap, run the following command:

```
$ kubectl create configmap occne-prometheus-alerts --from-file=/var/occne/cluster/<cluster-name>/artifacts/alerts -o yaml --dry-run -n occne-infra |
kubectl replace -f -
$ kubectl get configmap -n occne-infra
```

4. To verify the alerts in the Prometheus GUI, select the Alerts tab and view alert details by selecting any individual rule from the list of configured rules.

Configuring Alerts for CNE 1.9.0 and later versions

To configure BSF alerts in Prometheus for CNE 1.9.0, perform the following steps:

1. Copy the **BSF_Alertrules.yaml** file to the Bastion Host.
2. To create or replace the PrometheusRule CRD, run the following command:

```
$ kubectl apply -f ocbsf-alerting-rules.yaml -n <namespace>
```

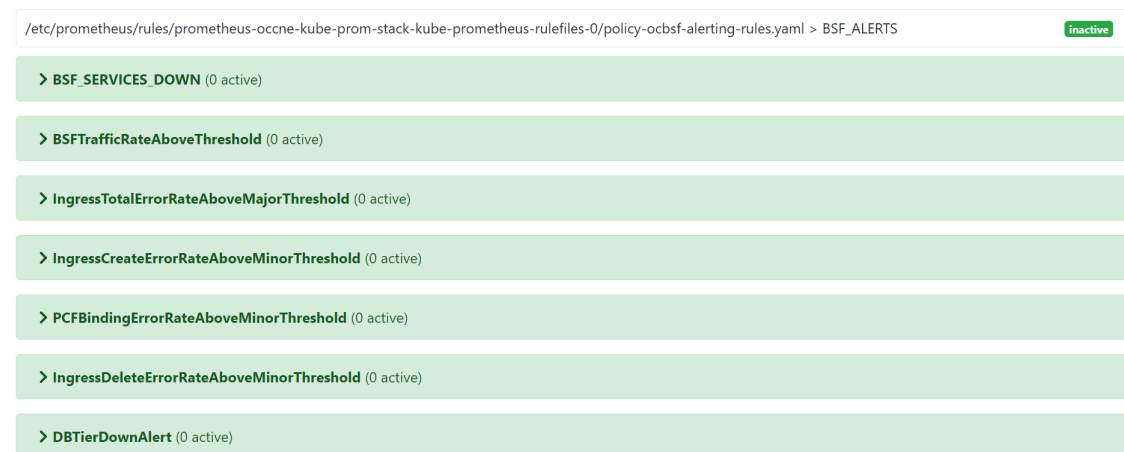
To verify if the CRD is created, run the following command:

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

3. To verify the alerts in the Prometheus GUI, select the Alerts tab and view alert details by selecting any individual rule from the list of configured alerts.

The following screen capture shows the Prometheus dashboard with BSF alerts configured for CNE 1.9.0:

Figure 2-1 BSF alerts on Prometheus Dashboard



Note

1. For upgrading to BSF 1.11.0 from a previous supported version on CNE 1.8.x, use the `BSF_Alertrules_cne1.5+.yaml` file. On the Prometheus dashboard, configure both old and new alert rules.
2. For installing BSF 1.11.0 on CNE 1.9.0 and later versions, use the `BSF_Alertrules_cne1.9+.yaml` file. On the Prometheus dashboard, configure only the new alert rules.

For the list of alerts supported in BSF, see *List of Alerts* section in *Oracle Communications Cloud Native Core, Binding Support Function User Guide*.

3

Customizing BSF

This chapter describes how to customize the Oracle Communications Cloud Native Core, Binding Support Function (BSF) deployment in a cloud native environment.

The BSF deployment is customized by overriding the default values of various configurable parameters in the `ocbsf_custom_values_26.1.201.yaml` file.

To customize the `ocbsf_custom_values_26.1.201.yaml` file as per the required parameters, perform the following steps:

1. Download the custom template from [My Oracle Support](#) (MOS). The custom template file is available in the software package.
2. Customize the file.
3. Save the updated file.

The BSF deployment is customized by overriding the default values of various configurable parameters in the `ocbsf_custom_values_26.1.201.yaml` file.

To customize the custom yaml file, perform the following steps:

1. Unzip `Custom_Templates` file available in the extracted documentation release package. For more information on how to download the package from [MOS](#), see [Downloading BSF Package](#) section.

The following files are used to customize the deployment parameters during installation:

- `ocbsf_custom_values_26.1.201.yaml`: This file is used to customize the deployment parameters of BSF.
 - `ocbsf_custom_values_servicemesh_config_26.1.201.yaml`: This file is used while configuring ASM Data Plane.
2. Save the updated files.

Note

- All parameters mentioned as mandatory must be present in the `ocbsf_custom_values_26.1.201.yaml` file.
- All fixed value parameters listed must be present in the custom-values yaml file with the exact values as specified in this section.

3.1 Configurations for Pre and Post Upgrade/Install Validations

This section describes mandatory configurable parameters that you must customize in the `ocbsf_custom_values_26.1.201.yaml` file for successful validation checks required on the application, databases, and related tables before and after BSF application upgrade/install.

Table 3-1 Configuration Parameter for Pre and Post Flight Checks

Parameter	Description	Mandatory(M)/ Optional(O) Parameter	Accepted values	Default Value
global.hookValidation.dbSchemaValidate	Specifies to perform database validations in case of pre-installation, pre-upgrade/post-upgrade/post-installation. Checks if the required databases and tables exist. Validates that the required columns exist in the tables and the correct foreign key exists (for config-server).	M	true/false	false Note: By default, this flag is false. In that case, validations is performed, and if the validation fails, a warning is logged and install/upgrade will continue. If this flag is true and the validation fails, an error is thrown and installation/upgrade fails.
global.operationalState	Specifies to control deployment operationalState, mainly during fault recovery set up installation in inactive mode, i.e., complete shutdown mode.	M	• NORMAL • PARTIAL_SHUTDOWN • COMPLETE_SHUTDOWN	&systemOperationalState NORMAL Note: Need to use this field along with enabling the field enableControlledShutdown as true

Table 3-1 (Cont.) Configuration Parameter for Pre and Post Flight Checks

Parameter	Description	Mandatory(M)/ Optional(O) Parameter	Accepted values	Default Value
global.hookValidation.infraValidate	Specifies to perform pre-flight infrastructure related validations like Replication Status, Critical Alerts, Kubernetes Version, and cnDbTier Version. Infrastructure related validations are done in the very beginning of the upgrade/install and if it fails, then install/upgrade will fail at this stage.	M	true/false	false Note: - Ensure helm parameters for replication Uri, dbTierVersionUri and alertmanagerUrl are pointing to working URI/URL respectively. - Before enabling infra Validate flag, ensure that there are no critical alarms exists before upgrading/ installing a new release in order to avoid failures. Also, make sure that replication is up.
appinfo.dbTierVersionUri	Specifies the URI provided by the db monitor service to query the cnDbTier Version. For example: http://mysql-cluster-db-monitor-svc.occne-cndbtier:8080/db-tier/version	M	URI	empty string

Table 3-1 (Cont.) Configuration Parameter for Pre and Post Flight Checks

Parameter	Description	Mandatory(M)/Optional(O) Parameter	Accepted values	Default Value
global.mysql.execution.ddlDelayTimeInMs	Adds a delay before the creation of configuration_item table, ensuring that topic_info table is created first and then the configuration_item table is created which has a foreign key dependency on topic_info. Specifies delay interval of 200 ms before inserting any entry into the ndb_replication table.	M	Interval in milliseconds	200 ms
appinfo.defaultReplicationStatusOnError	Specifies Replication Value in Case of any error on Infra Validation Replication Status	O	- UP - DOWN If the value is UP or empty string and the application throws an error while fetching replication status during infra-validation, the value of replication will be set as UP. If the value is DOWN, in case of any error while fetching replication status, the value of replication status will be set as DOWN.	UP
appinfo.cfgKeyViablePath	Specifies CfgKey of the microservice that will be used for extracting RELEASE_VERSION from database for validation of minimum Viable path.	O	It is CfgKey of the ReleaseConfig table.	public.hook.configserver

Table 3-1 (Cont.) Configuration Parameter for Pre and Post Flight Checks

Parameter	Description	Mandatory(M)/Optional(O) Parameter	Accepted values	Default Value
global.configWatcherVersion	Specifies the version of the config watcher. The version - V2 offers optimized polling of config server by services over V1.	O	V1, V2	V2

3.2 Configuring Mandatory Parameters

This section describes the mandatory configurable parameters that you must customize in the `ocbsf_custom_values_26.1.201.yaml` file for successful installation of Binding Support Function (BSF).

Table 3-2 Configurable Parameters for Mandatory Configurations

Parameter	Description
global.dockerRegistry	This mandatory parameter specifies the name of the Docker registry that hosts Binding Support Function docker images. Note: The Docker registry runs in OCCNE bastion server where all OAuth docker images are loaded.
global.nfInstanceId	This mandatory parameter specifies the unique NF InstanceID for each site deployed for BSF. To setup georedundancy, users must specify the value while deploying BSF; otherwise, georedundancy will not be supported. Default value for the parameter is UUID. Ensure nfInstanceId should be provided as UUID for fresh installation. On upgrade, the user should be using the original UUID or site Id which was provided during installation to avoid issues in upgrade. For more information, see Upgrading BSF . The same global nfInstanceId should be provided in appProfiles as well. The value of nfInstanceId must be unique for each site in a multi-site deployment.
global.envMySQLHost	This mandatory parameter specifies the IP address or host name of the MySQL server where BSF databases are hosted. Example: 10.196.33.106
global.envMySQLPort	This mandatory parameter specifies the port number of the MySQL server where BSF databases are hosted. Example: 3306
global.dbCredSecretName	This mandatory parameter specifies the name of the Kubernetes secret object that contains Database username and password. Default Value: ocbsf-db-pass
global.privilegedDbCredSecretName	This mandatory parameter specifies the name of the Kubernetes secret object containing Database username and password for an admin user. Default Value: ocbsf-privileged-db-pass

Table 3-2 (Cont.) Configurable Parameters for Mandatory Configurations

Parameter	Description
global.releaseDbName	This mandatory parameter specifies the name of the release database that contains details of release version. Default Value: ocbsf_release
global.serviceAccountName	Specifies the service account name if the service account, role, and rolebinding are created manually before the installation of BSF as per Creating Service Account, Role, and RoleBinding . Note serviceAccountName with empty quotes should be added in global section of all CV files.

Here is a sample configuration for mandatory parameters in ocbsf_custom_values_26.1.201.yaml file:

```
global:
  # Docker registry name
  dockerRegistry: ''
  nfInstanceId: "fe7d992b-0541-4c7d-ab84-c6d70b1b0666"
  # Primary MYSQL Host IP or Hostname
  envMysqlHost: &mysqlHostRef ''
  envMysqlPort: &mysqlPortRef ''
  # Jaeger hostname
  envJaegerAgentHost: ''
  # K8s secret object name containing OCBSF MYSQL UserName and Password
  dbCredSecretName: &dbCredSecretNameRef 'ocbsf-db-pass'
  privilegedDbCredSecretName: 'ocbsf-privileged-db-pass'
  #Release DB name containing release version details
  releaseDbName: 'ocbsf_release'
```

3.3 Enabling or Disabling Services Configurations

This section describes the configuration parameters that can be used to select the services that you want to enable or disable for your deployment.

To enable or disable the services, you must configure the following configurable parameters in the ocbsf_custom_values_26.1.201.yaml file:

Table 3-3 Configurable Parameters for Enabling/Disabling the BSF Core Service

Parameter	Description
global.bsfManagementEnable	This parameter determines if the BSF core service is enabled or not. Default Value: true
global.bsfManagementVersion1Enable	NA
global.bsfManagementVersion2Enable	NA

Note

The below section is applicable only when NF is required to register with NRF.

Table 3-4 Configurable Parameters for Enabling/Disabling the NRF Client Services

Parameter	Description
global.nrfClientNfManagementEnable	This is an optional parameter. Default Value: true
global.nrfClientNfDiscoveryEnable	This is an optional parameter. Default Value: false Note This parameter must be enabled only while using Oracle's BSF to PCF Binding Audit Retry feature.
global.appinfoServiceEnable	This optional parameter determines if the app info service is enabled or not. Default Value: true
global.performanceServiceEnable	This optional parameter determines if the performance service is enabled or not. Default Value: true

Table 3-5 Configurable Parameters for Enabling/Disabling the Diameter Gateway

Parameter	Description
global.diamGatewayEnable	This optional parameter determines if the diameter gateway is enabled or not. Default Value: true

Here is a sample configuration for configurable parameters in `ocbsf_custom_values_26.1.201.yaml` file:

```
global:
# BSF Core Services Enable/Disable option
  bsfManagementEnable: true
  bsfManagementVersion1Enable: false
  bsfManagementVersion2Enable: false

  nrfClientNfManagementEnable: true
  nrfClientNfDiscoveryEnable: true
  appinfoServiceEnable: true
  performanceServiceEnable: true
```

Table 3-6 Configurable Parameters for Enabling or Disabling BSF services for Query service

Parameter	Description
global.amServiceEnable	This optional parameter determines whether to enable query service for AM service. Default Value: false
global.smServiceEnable	This optional parameter determines whether to enable query service for SM service. Default Value: false
global.ueServiceEnable	This optional parameter determines whether to enable query service for UE service. Default Value: false
global.policydsEnable	This optional parameter determines whether to enable query service for Policy DS service. Default Value: false
global.pcrfCoreEnable	This optional parameter determines whether to enable query service for PCRF Core service. Default Value: false

Table 3-6 (Cont.) Configurable Parameters for Enabling or Disabling BSF services for Query service

Parameter	Description
global.bindingSvcEnabled	This optional parameter determines whether to enable query service for Binding service. Default Value: false

Table 3-7 Configurable Parameters to enable or disable the Audit Service

Parameter	Description
global.auditServiceEnable	Use this parameter to enable or disable audit service. Default Value: true
bsf-management-service.auditHandleNullAsStale	Specifies whether to handle null value as stale or not while auditing the tables of the service. Default Value: true
audit-service.recordsQueueCapacity	Specifies the number of stale records the queue can hold in an audit cycle. Default Value: 5K records if audit service configuration is based on <code>ocbsf_custom_values_minimal_26.1.201.yaml</code> file. 100K records if audit service configuration is based on <code>ocbsf_custom_values_26.1.201.yaml</code> file.
audit-service.maxTtlForceInterval	Specifies the grace interval (in seconds) after the expiry of Maximum TTL (Session Age) that is given to the service to delete an expired record gracefully. On expiry of this grace interval, Audit service will forcefully delete the records. Default Value: 259200 in minutes

3.4 Configuring Tracing Parameters

This section describes the configurable tracing parameters that you may customize in the `ocbsf_custom_values_26.1.201.yaml` file.

Table 3-8 Common Policy Configurable Parameters for OpenTelemetry

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Added/ Deprecat ed/ Updated in Release	Notes
envJaegerCollectorHost	Specifies the host direction where the Jaeger Collector is found.	Mandatory	occne-tracer-jaeger-collector.occne-infra	Added in Release 23.4.0	Make sure the jaeger Collector service is up and running inside OCCNE-Infra, with port specified in values.yaml
envJaegerCollectorPort	Specifies the port where the Jaeger Collector is listening to receive spans.	Mandatory	4318		Make sure this port matches with the one of your Jaeger Collector service port that is listening for OTLP formatted traces.
tracingEnabled	Specifies When 'true' enables the service to be instrumented by OpenTelemetry's Java Agent.	Mandatory	false		
tracingSamplerRatio	Specifies a ratio of spans which will be sent to the Jaeger Collector; i.e. of the total amount of spans, specify how many are going to be sent to the Jaeger Collector.	Mandatory	.001		Example: A value of "0.2" specifies that only 20 % of the spans are going to be sent. Range is 0 to 1.
tracingJdbcEnabled	Specifies when 'true' OpenTelemetry Java Agent will also show spans related to Database Operations.	Mandatory	false		If tracingEnabled is true on deployment, this will be enabled by default. In case tracingEnabled is false, this will also be false by default
tracingLogsEnabled	Specifies when 'true' enables spans and tracing logging	Mandatory	false		

Here is a sample configurations for OpenTelemetry in ocbfsf_custom_values_26.1.201.yaml file:

```
envJaegerCollectorHost: 'occne-tracer-jaeger-collector.occne-infra'
envJaegerCollectorPort: 4318 -> Make sure this matches with OCCNE-INFRA
jaeger collector service port.
tracing:
  tracingEnabled: 'true'
  tracingSamplerRatio: 0.001
  tracingJdbcEnabled: 'true'
  tracingLogsEnabled: 'false'
```

Table 3-9 Configurable Parameters for Tracing Configuration in Ingress Gateway

Parameter	Description
global.envJaegerAgentHost	This mandatory parameter specifies the Hostname or IP address for the jaeger agent. It is the FQDN of Jaeger Agent service running in OCCNE cluster under namespace occne-infra. It is written in the following format: <JAEGER_SVC_NAME>.<JAEGER_NAMESPACE>
global.envJaegerQueryUrl	This optional parameter specifies the query URL for the jaeger agent. Default Value: empty string
ingress-gateway.jaegerTelemetryTracingEnabled	This optional parameter specifies whether to enable or disable OpenTelemetry at Ingress Gateway. Default Value: false
ingress-gateway.openTelemetry.jaeger.httpExporter.host	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the host name of Jaeger collector host. Default Value: jaegercollector.cne-infra
ingress-gateway.openTelemetry.jaeger.httpExporter.port	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the port of Jaeger collector port. Default Value: 4318
ingress-gateway.openTelemetry.jaeger.probabilisticSampler	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the sampler where value is between 0.0 (no sampling) and 1.0 (sampling of every request). Default Value: 0.5

Here is a sample configuration for tracing in ingress-gateway in ocbsf_custom_values_26.1.201.yaml file:

```
jaegerTelemetryTracingEnabled: *tracingEnabled

openTelemetry:
  jaeger:
    httpExporter:
      host: *envJaegerCollectorHost
      port: *envJaegerCollectorPort
    probabilisticSampler: *tracingSamplerRatio
```

Table 3-10 Configurable Parameters for Tracing Configuration in Egress Gateway

Parameter	Description
egress-gateway.jaegerTelemetryTracingEnabled	This optional parameter specifies whether to enable or disable Jaeger Tracing at Egress Gateway. Default Value: true
egress-gateway.openTelemetry.jaeger.httpExporter.host	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the host name of Jaeger collector host. Default Value: jaegercollector.cne-infra

Table 3-10 (Cont.) Configurable Parameters for Tracing Configuration in Egress Gateway

Parameter	Description
egress-gateway.openTelemetry.jaeger.httpExporter.port	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the port of Jaeger collector port. Default Value: 4318
egress-gateway.openTelemetry.jaeger.probabilisticSampler	This is a mandatory paramter, if ingress-gateway.jaegerTelemetryTracingEnabled flag is set to true. It specifies the sampler where value is between 0.0 (no sampling) and 1.0 (sampling of every request). Default Value: 0.5

Here is a sample configuration for tracing in egress-gateway in ocsf_custom_values_26.1.201.yaml file:

```
jaegerTelemetryTracingEnabled: *tracingEnabled

openTelemetry:
  jaeger:
    httpExporter:
      host: *envJaegerCollectorHost
      port: *envJaegerCollectorPort
    probabilisticSampler: *tracingSamplerRatio
```

To configure tracing in nrf-client-nfdiscovery, you may configure the following configurable parameters in ocsf_custom_values_26.1.201.yaml file:

Table 3-11 Configurable Parameters for Tracing Configuration in nrfClientNfDiscovery

Parameter	Description
nrf-client.nrf-client-nfdiscovery.envJaegerSamplerParam	Note: You must customize this parameter only when NRF client services are enabled. Default Value: 1
nrf-client.nrf-client-nfdiscovery.envJaegerSamplerType	Note: You must customize this parameter only when NRF client services are enabled. Default Value: ratelimiting
nrf-client.nrf-client-nfdiscovery.envJaegerServiceName	Note: You must customize this parameter only when NRF client services are enabled. Default Value: pcf-nrf-client-nfdiscovery

Here is a sample configurations for tracing in ocsf_custom_values_26.1.201.yaml file:

```
nrf-client-nfdiscovery:
  envJaegerSamplerParam: '1'
  envJaegerSamplerType: ratelimiting
  envJaegerServiceName: pcf-nrf-client-nfdiscovery
```

To configure tracing in `nrf-client-nfmanagement`, you may configure the following configurable parameters in `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-12 Configurable Parameters for Tracing Configuration in nrf-client-nfmanagement

Parameter	Description
<code>nrf-client.nrf-client-nfmanagement.envJaegerSamplerParam</code>	Note: You must customize this parameter only when NRF client services are enabled. Default Value: 1
<code>nrf-client.nrf-client-nfmanagement.envJaegerSamplerType</code>	Note: You must customize this parameter only when NRF client services are enabled. Default Value: <code>ratelimiting</code>
<code>nrf-client.nrf-client-nfmanagement.envJaegerServiceName</code>	Note: You must customize this parameter only when NRF client services are enabled. Default Value: <code>pcf-nrf-client-nfmanagement</code>

Here is a sample configuration for tracing under `nrf-client-nfmanagement` in `ocbsf_custom_values_26.1.201.yaml` file:

```
nrf-client-nfmanagement:
  envJaegerSamplerParam: '1'
  envJaegerSamplerType: ratelimiting
  envJaegerServiceName: pcf-nrf-client-nfmanagement
```

To configure tracing in Alternate Route service, you should configure the following configurable parameters in `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-13 Configurable Parameters for Tracing Configuration in Alternate Route Service

Parameter	Description
<code>alternate-route.jaegerTracingEnabled</code>	Note: You must customize this parameter only when the alternate route service is enabled. Default Value: <code>false</code>
<code>alternate-route.openTracing.jaeger.udpSender.host</code>	Note: You must customize this parameter only when the alternate route service is enabled. Default Value: <code>occne-tracer-jaeger-agent.occne-infra</code>
<code>alternate-route.openTracing.jaeger.udpSender.port</code>	Note: You must customize this parameter only when the alternate route service is enabled. Default Value: 6831
<code>alternate-route.openTracing.jaeger.probabilisticSampler</code>	Note: You must customize this parameter only when the alternate route service is enabled. Default Value: 0.5

Here is a sample configurations for tracing in `ocbsf_custom_values_26.1.201.yaml` file:

```
jaegerTracingEnabled: true
openTracing :
  jaeger:
```

```

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

```

3.5 Configuring Database Names

This section describes the configuration parameters that can be used to customize the database names.

Note

Database name specified in the custom.yaml file should be used while creating the database during installation. See [Configuring Database, Creating Users, and Granting Permissions](#).

Table 3-14 Customizable Parameters for Database Name Configuration for BSF Services

Parameter	Description
bsf-management-service.envMysqlDatabase	This parameter specifies the name of the database of BSF Management Service. Default Value: ocpm_bsf
bsf-management-service.configserverContainerImage	This parameter specifies the name of the container image for Config Server. Note: Keep the image same as that of config server. Default Value: oc-config-server
bsf-management-service.envMysqlDatabaseConfigServer	This parameter specifies the name of the container image for Config Server. Default Value: ocbsf_config_server
bsf-management-service.envXnioTaskThreadPoolSize	This parameter specifies the number of XNIO Task threadpool size of BSF Management Service. Default Value: 180 Note: This parameter is not available in ocbsf_custom_values_26.1.201.yaml file. You must add this parameter to ocbsf_custom_values_26.1.201.yaml file with the given default value when required.

Table 3-14 (Cont.) Customizable Parameters for Database Name Configuration for BSF Services

Parameter	Description
config-server.envMysqlDatabase	This optional parameter specifies the name of the database for Config Server service. Default Value: ocbsf_config_server
cm-service.envCommonConfigMysqlDatabase	This optional parameter specifies the name of the database for CM service. Default Value: ocbsf_commonconfig
cm-service.envMysqlDatabase	This optional parameter specifies the name of the database for CM service. Default Value: ocbsf_cmsservice
cm-service.configserverContainerImage	This optional parameter specifies the container image name of Config Server service. Default Value: oc-config-server
cm-service.envMysqlDatabaseConfigServer	This optional parameter specifies the database name of Config Server service. Default Value: ocbsf_config_server
audit-service.envMysqlDatabase	This parameter specifies the name of the database for audit service. Default Value: ocbsf_audit_service
queryservice.envMysqlDatabase	This parameter specifies the name of the database for query service. Default Value: ocbsf_query
global.nrfClientDbName	This parameter specifies the name of the database of NRF Client. Default Value: ocbsf_nrf_client

Here is a sample configuration for configurable parameters in ocbsf_custom_values_26.1.201.yaml file:

```
global:
  nrfClientDbName: 'ocbsf_nrf_client'
bsf-management-service:
  envMysqlDatabase: 'ocpm_bsf'
  configserverContainerImage: oc-config-server
  envMysqlDatabaseConfigServer: 'ocbsf_config_server'

config-server:
```

```
envMySQLDatabase: ocbsf_config_server
```

```
cm-service:
```

```
envCommonConfigMySQLDatabase: ocbsf_commonconfig
envMySQLDatabase: ocbsf_cmsservice
configserverContainerImage: oc-config-server
envMySQLDatabaseConfigServer: 'ocbsf_config_server'
```

```
audit-service:
```

```
envMySQLDatabase: ocbsf_audit_service
```

Configuring Database Engine

The following table describes the parameter that you can configure to customize the default database engine used by BSF:

Table 3-15 Customizable Parameters for Database Engine for BSF

Parameter	Description	Notes
dbConfig.dbEngine	This mandatory parameter specifies the MySQL engine that is used by BSF to store information in the MySQL database. Default value: NDBCLUSTER	If the database engine is not NDBCLUSTER, then the value for this parameter can be changed only during fresh installation of BSF. Do not change the value of this parameter during upgrade scenarios.

Configuring NRF Client Multipod Feature

The following table describes the parameter that you can configure to customize the multipod support for NRF Client:

Table 3-16 Customizable Parameters for NRF Client Multipod Feature

Parameter	Description	Notes
nrf-client.nrf-client-nfmanagement.dbConfig.leaderPodDatabase	Specifies the database name for LeaderPodDb database. This database is unique per site. Default value: ocbsf_leaderPodDb	
nrf-client.nrf-client-nfmanagement.dbConfig.networkDbName	Specifies the release database name. Default value: ocbsf_release	

Configuring Database for Conflict Resolution

The following table describes the parameter that you can configure the database for conflict resolution in NDB cluster with replication:

Table 3-17 Customizable Parameters to enable or disable the Database for Conflict Resolution

Parameter	Description	Notes
global.mysql.conflictResolution.ndbConflictResolutionEnabled	This flag is used to prevent data conflicts in georeplicated deployments. When there are multiple sites with real-time replication, if a session is updated at both sites simultaneously, this is considered as a conflict. This flag configures the MySQL cluster replication to compare the updated timestamp in the session record, so the conflicts can be automatically resolved. In a single-site PCF, set this parameter to false. Note: This feature is only available if the database is MySQL cluster (NDB). For MySQL (innodb), the value for this flag must be set to false.	Even if its a single-site BSF NF deployment, set this parameter to true. As this will keep georedundancy and geo-replication enabled among the sites during multi-site deployment.
global.mysql.conflictResolution.useMaxDeleteWinInsConflictFn	This flag is used to update the Conflict Resolution Function to MAX_DEL_WIN_INS. Note: This feature is available if the NDB version is 8.0.30. If NDB version is less than 8.0.30, the value for this flag must be set to false.	

3.6 Configuring Database Indexing

The following table describes the parameters used to configure database indexing:

Table 3-18 Configurable parameters for database indexing

Parameter	Description	Mandatory (M)/Optional (O)/ Conditional (C)	Default Value	Added/ Updated/ Deprecated in release	Notes
performAddDropIndexAtSiteLevel	When this parameter is set to true, any index creation performed during a BSF upgrade occurs only at the local site and it is not replicated to the mate sites.	O	false	25.2.200	-

Note
This is applicable for Upgrade cases only

Table 3-18 (Cont.) Configurable parameters for database indexing

Parameter	Description	Mandatory (M)/Optional (O)/Conditional (C)	Default Value	Added/Updated/Deprecated in release	Notes
		n l y 			
bsfManagementService. metrics.altLookupTable.latency.percentiles	This parameter is used to configure the Percentiles of Latency metrics.	O	0.5, 0.9	26.1.200	This configuration can be provided either through deployment YAML or through custom values. Deployment YAML <pre>name: ALT_LOOKUP_TABLE_LATENCY_PERCENTILEV alue: {{.Values.metrics.altLookupTable.latency.percentiles quote }}</pre> Custom Values <pre>metrics: altLookupTable: latency: percentiles: '0.5, 0.9'</pre>

3.7 Configuring NRF client

This section describes the configurable parameters that you may customize for configuring NRF client. The configurations under `nrf-client` section allow Binding Support Function to register with NRF.

! Important

Before customizing parameters mentioned in this section, ensure that the NRF Client services are enabled by setting the value as `true` for `nrfClientNfManagementEnable`.

Table 3-19 Configurable Parameters for NRF Client Configuration

Parameter	Description
<code>global.nrfClientDbName</code>	This mandatory parameter that specifies the NRF Client database name. Default value: <code>ocbsf_nrf_client</code>
<code>global.deploymentNrfClientService.envNfNamespace</code>	This mandatory parameter that specifies the Kubernetes namespace of BSF.
<code>global.nrfClientCommonServicePort</code>	This mandatory parameter that specifies the port to be used for readiness and liveness probes. Default value: <code>9000</code>

Here is a sample configuration for NRF client in `ocbsf_custom_values_26.1.201.yaml` file:

```
global:

  nrfClientDbName: 'ocbsf_nrf_client'

  deploymentNrfClientService:
    envEgressGatewayPort: *svcEgressGatewayHttp
    #K8s namespace of BSF
    envNfNamespace: ''
    #same as bsfApiRoot
    nfApiRoot: ''

  nrfClientCommonServicePort: *containerMonitoringHttp
```

Table 3-20 Configurable Parameters for NRF Client Configuration

Parameter	Description
<code>nrf-client.configmapApplicationConfig</code>	This mandatory parameter is used to provide inputs to NRF-Client.
<code>&configRef</code>	This mandatory reference variable is used to take the input from the config map.
<code>nrf-client.configmapApplicationConfig.profile</code>	This mandatory parameter specifies the NF profile of BSF that is registered with NRF. See config-map table for more details.

Table 3-20 (Cont.) Configurable Parameters for NRF Client Configuration

Parameter	Description
appinfo.infraServices	Specifies the URI for the health check of InfraServices that need to be monitored. Examples: <code>http://mysql-cluster-db-monitor-svc.vzw1-cndbtier:8080/actuator/health</code> <code>http://mysql-cluster-db-replication-svc.vzw1-cndbtier/actuator/health</code> Uncomment this parameter and set this parameter to an empty array if any one of following conditions is true: - Deploying on OCCNE 1.4 or lower version - Not deploying on OCCNE - Do not wish to monitor infra services such as db-monitor service This parameter uses the default namespace - <code>occne-infra</code>. If <code>cnDBTier</code> is used to deploy BSF, this field must be updated accordingly.
appinfo.core_services.bsf	Specifies the list of BSF services to be monitored.
appinfo.core_services.common	Specifies the list of common services to be monitored.
perf-info.configmapPerformance.prometheus	This conditional parameter specifies the Prometheus server URL. Default Value: <code>http://prometheus-server.prometheus:5802</code> <code>jaeger=jaeger-agent.occne-infra</code> <code>jaeger_query_url=http://jaeger-query.occne-infra</code> Note: If you do not specify any value for this parameter, BSF reports 0 loads to NRF.

Configurable parameters for NRF Client Configuration in Config-map

Parameter	Description
primaryNrfApiRoot	Primary NRF hostname and port in the following format: <code><http scheme>://<Hostname/IP>:<Port></code> This parameter can only contain valid API root. For example: <code>http://nrf1-api-gateway.svc:80</code>
secondaryNrfApiRoot	Secondary NRF hostname and port in the following format: <code><http scheme>://<Hostname/IP>:<Port></code> This parameter can only contain valid API root. For example: <code>http://nrf2-api-gateway.svc:80</code>
retryAfterTime	When primary NRF is down, this will be the wait Time (in ISO 8601 duration format) after which request to primary NRF will be retried to detect primary NRF's availability. This parameter can only contain valid ISO 8601 duration format. For example: <code>PT120S</code>

Parameter	Description
nrfClientType	The NfType of the NF registering. The value for this parameter must be set to BSF.
nrfClientSubscribeTypes	Network functions for which BSF wants to discover and subscribe to the NRF.
appProfiles	NfProfile of BSF to be registered with NRF. This parameter can only contain valid NF profile. During fresh install or upgrade, the value of this parameter is loaded into the database and then used to trigger NfRegister or NfUpdate operation to NRF. For any subsequent changes to appProfile, REST API or CNC Console must be used. For more information, see <i>Oracle Communications Cloud Native Core Binding Support Function REST Specification Guide</i> or <i>Oracle Communications Cloud Native Core Binding Support Function User Guide</i>. Note: It is a 3GPP defined data type. To know more about its attributes, refer to <i>3GPP TS 29.510 version 16.4.0 Release 16</i>.
enableF3	Support for 29.510 Release 15.3 This parameter can only have true (default) or false as values.
enableF5	Support for 29.510 Release 15.5 This parameter can only have true (default) or false as values.
renewalTimeBeforeExpiry	Time Period (in seconds) before the Subscription Validity time expires. For example: 3600
validityTime	The default validity time (in days) for subscriptions. For example: 30
enableSubscriptionAutoRenewal	This parameter can be used to enable renewal of subscriptions automatically. This parameter can only have true (default) or false as values.
nfHeartbeatRate	This parameter specifies the rate at which BSF shall heartbeat with the NRF. The value shall be configured in terms of percentage (1-100). If the heartbeatTimer is 60s, then the NF shall heartbeat at $\text{nfHeartBeatRate} * 60/100$. Default Value: 80
acceptAdditionalAttributes	This parameter can be used to enable additional Attributes as part of 29.510 Release 15.5. This parameter can only have true or false (default) as values.
enableVirtualNrfResolution	This parameter can be used to enable or disable virtual NRF session retry by Alternate routing service. This parameter can only have true or false (default) as values.
virtualNrfFqdn	This parameter specifies the virtual NRF FQDN being used to query static list of route. By default, the value for this parameter is set to false.
virtualNrfScheme	This parameter specifies the scheme to be used with the virtual Fqdn. By default, the value for this parameter is set to http.
virtualNrfPort	This parameter specifies the port to be used with the virtual Fqdn. By default, the value for this parameter is set to 8080.
requestTimeoutGracePeriod	An additional grace period where no response is received from the NRF. This additional period shall be added to the requestTimeout value. It ensures that the egress-gateway shall first timeout, and send an error response to the NRF-client. Default Value: 2
nrfRetryConfig	It specifies the configurations required for the NRF Retry mechanism.
healthCheckConfig	It specifies the configurations required for the Health check of NRFs.

Parameter	Description
serviceRequestType	Specifies the type of service request.
primaryNRFRetryCount	Specifies the number of times a service request is retried to the primary NRF in case of failure.
nonPrimaryNRFRetryCount	Specifies the number of times a service request is retried to the non-primary NRF in case of failure.
alternateNRFRetryCount	Specifies the number of alternate NRFs that are retried in case of failure. When the value is specified as -1, all available NRF instances are tried.
errorReasonsForFailure	Specifies the HTTP status codes or exceptions for which retry is attempted.
gatewayErrorCodes	Specifies the HTTP status codes sent by the Egress Gateway for which retry is attempted.
requestTimeout	Specifies the timeout period where no response is received from the Egress Gateway.

Parameter	Description
healthCheckCount	Specifies the number of consecutive success or failures responses required to mark an NRF instance healthy or unhealthy.
healthCheckInterval	Specifies the interval at which a health check of an NRF is performed.
requestTimeout	Specifies the timeout period where no response is received from the Egress Gateway.
errorReasonsForFailure	Specifies the HTTP status codes or exceptions for which retry is attempted.
gatewayErrorCodes	Specifies the HTTP status codes sent by the Egress Gateway for which retry is attempted.

Here is a sample configuration for NRF client in `ocbsf_custom_values_26.1.201.yaml` file:

```
deploymentNrfClientService:
  #K8s namespace of BSF
  envNfNamespace: ''

appinfo:
  serviceAccountName: ''
  # Set Infrastructure services to empty array if any one of below condition
  # is met
  # 1. Deploying on occne 1.4 or lesser version
  # 2. Not deploying on OCCNE
  # 3. Do not wish to monitor infra services such as db-monitor service
  # then the below mentioned attribute 'infra_services' should be uncommneted
  # and epmty array should be passed as already mentioned.
  #infraServices: []

perf-info:
  configmapPerformance:
    prometheus: ''

nrf-client:
  # This config map is for providing inputs to NRF-Client
```

```

configmapApplicationConfig:
  &configRef
  # primaryNrfApiRoot - Primary NRF Hostname and Port
  # SecondaryNrfApiRoot - Secondary NRF Hostname and Port
  # retryAfterTime - Default downtime(in ISO 8601 duration format) of an
NRF detected to be unavailable.
  # nrfClientType - The NfType of the NF registering
  # nrfClientSubscribeTypes - the NfType for which the NF wants to
subscribe to the NRF.
  # appProfiles - The NfProfile of the NF to be registered with NRF.
  # enableF3 - Support for 29.510 Release 15.3
  # enableF5 - Support for 29.510 Release 15.5
  # renewalTimeBeforeExpiry - Time Period(seconds) before the Subscription
Validity time expires.
  # validityTime - The default validity time(days) for subscriptions.
  # enableSubscriptionAutoRenewal - Enable Renewal of Subscriptions
automatically.
  # nfHeartbeatRate - This value specifies the rate at which the NF shall
heartbeat with the NRF. The value shall be configured in terms of
percentage(1-100). if the heartbeatTimer is 60s, then the NF shall heartbeat
at nfHeartBeatRate * 60/100
  # acceptAdditionalAttributes - Enable additionalAttributes as part of
29.510 Release 15.5
  # enableVirtualNrfResolution=false
  # virtualNrfFqdn=nf1stub.ocpcf.svc
  # virtualNrfScheme=http
  # virtualNrfPort=8080
  # requestTimeoutGracePeriod=2
  # nrfRetryConfig=[{ "serviceRequestType": "ALL_REQUESTS",
"primaryNRFRetryCount": 1, "nonPrimaryNRFRetryCount" : 1,
"alternateNRFRetryCount" : -1, "errorReasonsForFailure":
[503,504,500,"SocketTimeoutException","JsonProcessingException","UnknownHostEx
ception","NoRouteToHostException", "IOException"], "gatewayErrorCodes":
[503,429], "requestTimeout": 100 },{"serviceRequestType":
"AUTONOMOUS_NFREGISTER", "primaryNRFRetryCount": 1,
"nonPrimaryNRFRetryCount": 1, "alternateNRFRetryCount": -1,
"errorReasonsForFailure":
[503,504,500,"SocketTimeoutException","JsonProcessingException","UnknownHostEx
ception","NoRouteToHostException", "IOException"], "gatewayErrorCodes":
[503,429], "requestTimeout": 100 }]
  # healthCheckConfig={ "healthCheckCount": -1, "healthCheckInterval": 5,
"requestTimeout": 10, "errorReasonsForFailure":
[503,504,500,"SocketTimeoutException","JsonProcessingException","UnknownHostEx
ception","NoRouteToHostException", "IOException"], "gatewayErrorCodes":
[503,429] }
  profile: |-
    [appcfg]
    primaryNrfApiRoot=nrf1-api-gateway.svc:80
    secondaryNrfApiRoot=nrf2-api-gateway.svc:80
    nrfScheme=http
    retryAfterTime=PT120S
    nrfClientType=BSF
    nrfClientSubscribeTypes=
    appProfiles= [{"nfInstanceId":"fe7d992b-0541-4c7d-ab84-
c6d70b1b0666","nfType":"BSF","nfStatus":"REGISTERED","fqdn":"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com","priority":1,"capac

```

```

ity":1,"load":2,"bsfInfo":{"ipv4AddressRanges":
[{"start":"10.0.0.1","end":"10.113.255.255"}], "ipv6PrefixRanges":
[{"start":"2800:a00:cc03::/64","end":"2800:a00:cc04::/64"}]}, "nfServices":
[{"serviceInstanceId":"03063893-
cf9e-4f7a-9827-111111111111","serviceName":"nbsf-management","versions":
[{"apiVersionInUri":"v1","apiFullVersion":"1.0.0"}], "scheme":"http","nfService
Status":"REGISTERED","fqdn":"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com","interPlmnFqdn":nul
l,"ipEndpoints":
[{"ipv4Address":"10.233.22.149","transport":"TCP","port":80}], "apiPrefix":null
,"allowedNfTypes":["PCF","AF","NEF"],"priority":1,"capacity":1,"load":2}}]
enableF3=true
enableF5=true
renewalTimeBeforeExpiry=3600
validityTime=30
enableSubscriptionAutoRenewal=true
nfHeartbeatRate=80
acceptAdditionalAttributes=false
enableVirtualNrfResolution=false
virtualNrfFqdn=nf1stub.ocpcf.svc
virtualNrfScheme=http
virtualNrfPort=8080
requestTimeoutGracePeriod=2
nrfRetryConfig=[{ "serviceRequestType": "ALL_REQUESTS",
"primaryNRFRetryCount": 1, "nonPrimaryNRFRetryCount" : 1,
"alternateNRFRetryCount" : -1, "errorReasonsForFailure":
[503,504,500,425,"SocketTimeoutException","JsonProcessingException","UnknownHo
stException","NoRouteToHostException","IOException"], "gatewayErrorCodes":
[503,429,425], "requestTimeout": 10 },{"serviceRequestType":
"AUTONOMOUS_NFREGISTER", "primaryNRFRetryCount": 1,
"nonPrimaryNRFRetryCount": 1, "alternateNRFRetryCount": -1,
"errorReasonsForFailure":
[503,504,500,425,"SocketTimeoutException","JsonProcessingException","UnknownHo
stException","NoRouteToHostException","IOException"], "gatewayErrorCodes":
[503,429,425], "requestTimeout": 10 }]
healthCheckConfig={ "healthCheckCount": -1, "healthCheckInterval": 5,
"requestTimeout": 10, "errorReasonsForFailure":
[503,504,500,425,"SocketTimeoutException","JsonProcessingException","UnknownHo
stException","NoRouteToHostException","IOException"], "gatewayErrorCodes":
[503,429,425] }

```

Table 3-21 Configurable Parameters for nrf-client-nfdiscovery

Parameter	Description
nrf-client.nrf-client-nfdiscovery.configmapApplicationConfig	This mandatory parameter is used to provide inputs to NRF Client for NF discovery.
nrf-client.nrf-client-nfdiscovery.readinessProbe.httpGet.port	This mandaroty parameter that specifies the port to be used for readiness probes. Default Value: 9000
nrf-client.nrf-client-nfdiscovery.livenessProbe.httpGet.port	This mandaroty parameter that specifies the port to be used for liveness probes. Default Value: 9000

Here is a sample configuration for NRF client in `ocbsf_custom_values_26.1.201.yaml` file:

```
nrf-client:
  nrf-client-nfdiscovery:
    readinessProbe:
      httpGet:
        port: *containerMonitoringHttp
    livenessProbe:
      httpGet:
        port: *containerMonitoringHttp
```

Table 3-22 Configurable Parameters for nrf-client-nfmanagement

Parameter	Description
<code>nrf-client.nrf-client-nfmanagement.configmapApplicationConfig</code>	This mandatory is used to provide inputs to NRF Client for NF management.
<code>nrf-client.nrf-client-nfmanagement.readinessProbe.httpGet.port</code>	This mandatory parameter that specifies the port to be used for readiness probes. Default Value: 9000
<code>nrf-client.nrf-client-nfmanagement.livenessProbe.httpGet.port</code>	This mandatory parameter that specifies the port to be used for liveness probes. Default Value: 9000

Here is a sample configuration for NRF client in `ocbsf_custom_values_26.1.201.yaml` file:

```
nrf-client:
  nrf-client-nfmanagement:
    readinessProbe:
      httpGet:
        port: *containerMonitoringHttp
    livenessProbe:
      httpGet:
        port: *containerMonitoringHttp
```

3.8 Configuring Diameter Gateway

This section describes the configurable parameters that you may customize for configuring diameter gateway,

① Note

You must configure the parameters listed in the following table only when diameter gateway is enabled.

Table 3-23 Configurable Parameters for Diameter Gateway

Parameter	Description
diam-gateway.diamResponseLatencyHistogramBucket	Latency bucket values for the diameter response histogram latency metrics - - ocbsf_diam_response_latency_seconds_bucket - ocbsf_diam_service_overall_processing_time_seconds_bucket Default values: 10,20,30,40,60,80,100 Note:With the above <i>diamResponseLatencyHistogramBucket</i> configured values, the Micrometer automatically creates additional buckets (1, 2, 3, 4, 6, 8) beyond what's configured. This is done to ensure that all possible values are accounted for and provide the accuracy of the metrics.
diam-gateway.diamResponseLatencyHistogramPercentile	Latency percentile values for the diameter response histogram latency metrics- - ocbsf_diam_response_latency_seconds - ocbsf_diam_service_overall_processing_time_seconds Default values: 0.50, 0.90, 0.95, 0.99
diam-gateway.configserverContainerImage	This parameter specifies the name of the container image of configuration service for Diameter Gateway. For BSF, the default value is set to oc-config-server.
diam-gateway.envMySQLDatabaseConfigServer	This parameter specifies the name of the database of configuration service for Diameter Gateway. For BSF, the default value is set to ocbsf_config_server.
diam-gateway.envDiameterRealm	This mandatory parameter specifies the Diameter Realm of BSF diameter gateway. For example, oracle.com.
diam-gateway.envDiameterIdentity	This mandatory parameter specifies the Diameter host of BSF diameter gateway. For example, ocbsf-diam-gateway.
diam-gateway.envDbConnStatusHttpEnabled	This parameter specifies whether to enable or disable monitoring of the connectivity status of the database service. The default value of this parameter is false.
diam-gateway.envDiameterHostIp	Contains all the k8s cluster worker node names and corresponding IP addresses in the following format: NodeName1=<ip1>,NodeName2=<ip2> If LoadBalancer is being used, provide its IP.

Here is a sample configuration in ocbsf_custom_values_26.1.201.yaml file:

```
diam-gateway:
  configserverContainerImage: *configServerImage
  envMySQLDatabaseConfigServer: *configServerDB

  # Graceful Termination
  gracefulShutdown:
    gracePeriod: 30s
  envDiameterRealm: 'oracle.com'
  envDiameterIdentity: 'ocbsf-diam-gateway'
  #This should contain all the k8s cluster worker node name and ip
  #corresponding to it in a format i.e. NodeName1=<ip1>,NodeName2=<ip2>
  #If LoadBalancer is being used then give all ip as LoadBalancer's ip
  envDiameterHostIp: ''
```

```

envDbConnStatusHttpEnabled: false
staticIpAddress: ''
staticDiamNodePort: *svcDiamGatewayDiamNodePort
deployment:
  customExtension:
    annotations: {
      # Enable this section for service-mesh based installation
      #   traffic.sidecar.istio.io/excludeOutboundPorts: "9000,5801",
      #   traffic.sidecar.istio.io/excludeInboundPorts: "9000,5801"
    }

```

The `lbService` provides the annotations and labels for service diameter gateway and the `nonlbService` provides annotations and labels for headless diameter gateway.

3.9 API Root Configuration for Notification URI

This section describes the configuration parameters that can be used to API Root configuration.

To configure these parameters, you should configure the following configurable parameters in the `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-24 Configurable Parameters for Api Root Configuration for Notification URI

Parameter	Description
<code>global.bsfcApiRoot</code>	This optional parameter specifies the API root of BSF that is used in notification URLs generated by BSF's when sending request to other producer NFs. If the value is not configured for this parameter, the ingress gateway service name and port is used as default value. For example: <code>https://<Helm namespace>-ocbsf-ingress-gateway:443</code> .
<code>global.deploymentNrfClientService.nfApiRoot</code>	This mandatory parameter specifies Api root of BSF. Note: This parameter must be configured only when when NRF Client services are enabled. Its value should be same as the value of " <code>global.bsfcApiRoot</code> " parameter.

API root of BSF that will be used in notification URLs generated by BSF's when sending request to other producer NFs

#If not configured then the ingress gateway service name and port will be used as default value. ex:"https://<helm name>-ocbsf-ingress-gateway:443"

```

global:
  bsfcApiRoot: ''
  deploymentNrfClientService:
    #same as bsfcApiRoot
    nfApiRoot: ''

```

3.10 TLS Configurations

The following table describes the newly introduced and updated parameters for TLS:

Table 3-25 Helm Parameters

Parameter Name	Description	Mandatory/ Optional/ Conditional	Details
clientDisabledExtension	Disables the extension sent by messages originated by clients (ClientHello).	O	Data Type: String Range: NA Default Value: ec_point_formats
serverDisabledExtension	Disables the extension sent by messages originated by servers (ServerHello).	O	Data Type: String Range: NA Default Value: null
tlsNamedGroups	Provides a list of values sent in the supported_groups extension. These are comma-separated values.	O	Data Type: String Range: NA Default Value: null
clientSignatureSchemes	Provides a list of values sent in the signature_algorithms extension. These are comma-separated values.	O	Data Type: String Range: NA Default Value: null
tlsVersion	Indicates the TLS version.	M	Data Type: String Range: • TLSv1.2, TLSv1.3 • TLSv1.2 • TLSv1.3 Default Value: TLSv1.2, TLSv1.3
allowedCipherSuites	Indicates allowed Ciphers suites.	O	Data Type: String Range: NA Default Values: • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256

Table 3-25 (Cont.) Helm Parameters

Parameter Name	Description	Mandatory/ Optional/ Conditional	Details
cipherSuites	Indicates supported cipher suites.	O	Data Type: String Range: NA Default Values: • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256

The following global sample Helm configuration is required for TLS 1.3:

```
global:
  tlsVersion: &tlsVersion 'TLSv1.2,TLSv1.3'
  # supportedCipherSuiteList: &supportedCipherSuiteList
  'TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256'
  cipherSuites: &cipherSuites
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
    - TLS_AES_128_GCM_SHA256
    - TLS_CHACHA20_POLY1305_SHA256

  allowedCipherSuites: &allowedCipherSuites
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
    - TLS_AES_128_GCM_SHA256
    - TLS_CHACHA20_POLY1305_SHA256
```

The following Egress Gateway configuration is required for TLS 1.3:

```
egress-gateway:
#Cipher Suites to be enabled on client side
  clientDisabledExtension: null
  serverDisabledExtension: null
  tlsNamedGroups: null
  clientSignatureSchemes: null

#Enabling it for egress https requests
  enableOutgoingHttps: true

#Enabling it for egress http1.1 requests
  http1:
    enableOutgoingHTTP1: false # Flag to enable or disable the feature

egressGwCertReloadEnabled: false
egressGwCertReloadPath: /egress-gw/store/reloadservice:
  ssl:
    tlsVersion: *tlsVersion
    #supportedCipherSuiteList:*supportedCipherSuiteList
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      ecdsa:
        fileName: ssl_ecdsa_private_key.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: ocegress.cer
      ecdsa:
        fileName: ssl_ecdsa_certificate.crt
    caBundle:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: caroot.cer
    keyStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: key.txt
    trustStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: trust.txt

httpsTargetOnly: "true"
```

```
#true: Means change Scheme of RURI to http
#false: Keep scheme as is.
httpRuriOnly: "false"
```

Note

"httpsTargetOnly" must be set to true and "httpRuriOnly" must be set to false.

The following Ingress Gateway configuration is required for TLS 1.3:

```
ingress-gateway:
#Cipher Suites to be enabled on client side
  clientDisabledExtension: null
  serverDisabledExtension: null
  tlsNamedGroups: null
  clientSignatureSchemes: null

# Enable it to accept incoming http requests
enableIncomingHttp: true

# ---- HTTPS Configuration - BEGIN ----
enableIncomingHttps: true

service:
  ssl:
    tlsVersion: *tlsVersion
    #supportedCipherSuiteList: *supportedCipherSuiteList
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: ocbgress.cer
    caBundle:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: caroot.cer
    keyStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: key.txt
    trustStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: trust.txt
```

For more information on HTTPS Configurations in Egress/Ingress Gateway, see [Configuring Ingress/Egress Gateway HTTPS](#).

The following NRF configuration is required for TLS 1.3:

```
nrf-client:
  profile: |-
    [appcfg]
    primaryNrfApiRoot=nrf1-api-gateway.svc:8443
    secondaryNrfApiRoot=nrf2-api-gateway.svc:8443
    nrfScheme=https
```

Note

NRF ports must be changed from 8080 to 8443. Moreover, the nrfScheme must be changed from http to https.

3.11 TLS Configuration in Diameter Gateway

Configurable Parameters for TLS in Diameter Gateway:

Table 3-26 Configurable Parameters for TLS in Diameter Gateway

Parameter	Description	Mandatory/Optional/Conditional	Default Value
TLS_ENABLED	To enable or disable TLS.	O	false
TLS_DIAMETER_PORT	Listening port diameter TLS.	O	5868

Table 3-26 (Cont.) Configurable Parameters for TLS in Diameter Gateway

Parameter	Description	Mandatory/Optional/Conditional	Default Value
TLS_CIPHER_SUITE	To configure ciphers suites.	O	TLS 1.2 • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS 1.3 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256
TLS_INITIAL_ALGORITHM	To configure initial algorithm. ES256 or RS256	O	RS256
TLS_SECRET_NAME	Secret name for TLS configs	O	dgw-tls-secret
TLS_RSA_PRIVATE_KEY_FILENAME	To configure the filename for RSA private key, that will be stored in secret.	O	dgw-key.pem
TLS_ECDSA_PRIVATE_KEY_FILENAME	To configure the filename for ECDSA private key, that will be stored in secret.	O	dgw-ecdsa-private-key.pem
TLS_RSA_CERTIFICATE_FILENAME	To configure the filename for RSA certificate, that will be stored in secret.	O	dgw-cert.crt
TLS_ECDSA_CERTIFICATE_FILENAME	To configure the filename for ECDSA certificate, that will be stored in secret.	O	dgw-ecdsa-certificate.crt
TLS_CA_BUNDLE_FILENAME	To configure the filename for CA Bundle, that will be stored in secret.	O	ca-cert.cer
TLS_MTLS_ENABLED	To enable or disable mTLS	O	true

Note

Selective enabling of TLS version can be done through Diameter Gateway deployment file. The TLS_VERSION can be

- TLSv1.2, TLSv1.3
- TLSv1.2
- TLSv1.3

The following global sample Helm configuration is required for TLS in Diameter Gateway:

```
tls:
  enabled: false
  initialAlgorithm: 'RS256'
  secretName: 'dgw-tls-secret'
  cipherSuites:
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
    - TLS_AES_128_GCM_SHA256
    - TLS_CHACHA20_POLY1305_SHA256
```

3.12 Configuration for TLS Communication between Pod and Kubernetes API Server

Configurable Parameters for TLS communication between pod and kubeApi:

Table 3-27 Configurable Parameters for TLS Communication between Pod and kubeApi

Parameter	Description	Mandatory/Optional/Conditional	Default Value
tlsVersionSupportForKubeApiServer.enabled	To enable or disable TLS for Kubernetes API communication	O	False
tlsVersionSupportForKubeApiServer.kubeApiServerTlsVersion	If the flag is enabled, user can choose TLSv1.3 or TLSv1.2 for communication between each service and Kubernetes API server.	O	TLSv1.2

Table 3-27 (Cont.) Configurable Parameters for TLS Communication between Pod and kubeApi

Parameter	Description	Mandatory/Optional/Conditional	Default Value
tlsVersionSupportForKubeApiServer.cipherSuites	Indicates supported cipher suites.	O	Default Values for TLSV1.3: • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256 Default Values for TLSV1.2: • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
tlsVersionSupportForKubeApiServer.featureSecrets	Indicates the feature secrets needed to enable the TLS 1.3	O	- ocpcf-gateway-secret - ocegress-secret-sasl Note: If you need to change the default values, you must modify the secret names accordingly.
global.kubeApiServerTlsEnabled	Enable or disable the TLS and SSL context and configurations for communication between pod and kubeApi.	O	True

Table 3-27 (Cont.) Configurable Parameters for TLS Communication between Pod and kubeApi

Parameter	Description	Mandatory/Optional/Conditional	Default Value
global.kubeApiServerTlsVersion	To set the TLS version for communication between pod and kubeApi. If the flag is enabled, user can choose TLSv1.3 or TLSv1.2 for communication between each service and Kubernetes API server.	O	TLSv1.2 Possible values: TLSv1.2 or TLSv1.3
global.kubeApiServerTlsCipherSuites	Indicates the cipher suites for TLS version	O	# if TLSv1.2 is configured, comment out the first 3 TLSv1.3 ciphers and uncomment the other TLSv1.2 ciphers # - TLS_AES_256_GCM_SHA384 # - TLS_AES_128_GCM_SHA256 # - TLS_CHACHA20_POLY1305_SHA256 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

The following sample Helm configuration is required for TLS Kubernetes API Server:

```
# ----tlsVersionSupportForKubeApiServer CONFIGURATION - BEGIN ----
tlsVersionSupportForKubeApiServer:
  enabled: false
  kubeApiServerTlsVersion: *kubeApiServerTlsVersion # Should be either
  TLSv1.3 or TLSv1.2, comma separated values is not supported
  cipherSuites: # if TLSv1.2 is configured, comment out the first 3 TLSv1.3
  ciphers and uncomment the other TLSv1.2 ciphers
```

```

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256
# - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
# - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
# - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
# - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
# - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
# List of all the secrets to be volume mounted
featureSecrets: # When the above feature flag is enabled, configured
secrets in all the features shall be configured below, otherwise helm upgrade/
install will fail.
- ocpcf-gateway-secret
- ogress-secret-sasl
# ----tlsVersionSupportForKubeApiServer CONFIGURATION - END ----

```

Note

These configuration are required to be done at Ingress Gateway and Egress Gateway.

The following global sample Helm configuration is required for TLS Kubernetes API Server in BSF:

```

kubeApiServerTlsVersion: &kubeApiServerTlsVersion TLSv1.2s # Should be either
TLSv1.3 or TLSv1.2, comma separated values is not
supported

```

3.13 Configuring Ingress Gateway

This section describes the configuration parameters that are required for configurations in Ingress Gateway.

Note

Following configurations are applicable only when Ingress Gateway is enabled.

Table 3-28 Configurable Parameters for Ingress Gateway at Global Section

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.metalLbpAllocationEnabled	Enable or disable IP Address allocation from MetalLB Pool	Optional	false
global.metalLbpAllocationAnnotation	Address Pool Annotation for MetalLB	Optional	metallb.universe.tf/ address-pool: signaling
global.tlsVersion	Indicates the TLS version	Mandatory	TLSv1.2, TLSv1.3

Table 3-28 (Cont.) Configurable Parameters for Ingress Gateway at Global Section

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.allowedCipherSuites	Indicates allowed Ciphers suites	Optional	• TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256
global.CipherSuites	Indicates supported cipher suites	Optional	• TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256

Table 3-29 Configurable Parameters for Ingress Gateway

Parameter	Description	Mandatory /Optional Parameter	Default Value
ingress-gateway.enableIncomingHttp	Enable it to accept incoming http requests	Optional	true

Table 3-29 (Cont.) Configurable Parameters for Ingress Gateway

Parameter	Description	Mandatory /Optional Parameter	Default Value
ingress-gateway.ingressServer.keepAlive.enabled		Optional	false
ingress-gateway.ingressServer.keepAlive.idealTime		Optional	180 (in seconds)
ingress-gateway.ingressServer.keepAlive.count		Optional	9
ingress-gateway.ingressServer.keepAlive.interval		Optional	60 (in seconds)
ingress-gateway.minAvailable	Specifies the number of pods that must always be available, even during a disruption.	Optional	1
ingress-gateway.minReplicas	Specifies the minimum replicas to scale to maintain an average CPU utilization.	Optional	1
ingress-gateway.maxReplicas	Specifies the maximum replicas to scale to maintain an average CPU utilization.	Optional	1
ingress-gateway.userAgentHeaderValidationConfigMode	This flag is used to accept the user-agent configurations from Helm or REST.	Mandatory	Helm
ingress-gateway.userAgentHeaderValidation.enabled	Specifies the type of validation that will be taken into consideration when processing the values born on the user agent header.	Mandatory	False
ingress-gateway.userAgentHeaderValidation.validationType	Specifies the type of validation that will be taken into consideration when processing the values on the user agent header.	Mandatory	Relaxed
ingress-gateway.userAgentHeaderValidation.consumerNfTypes	Compares the NF Type born in the user agent header present in the incoming requests towards CNC PCF's Ingress Gateway.	Mandatory	EMPTY
ingress-gateway.enableIncomingHttps	To enable HTTPS for ingress traffic.	Mandatory	false
ingress-gateway.service.ssl.privateKey.k8SecretName	Name of the Kubernetes Secret which contains the private key for BSF,	Mandatory	Not applicable
ingress-gateway.service.ssl.privateKey.k8Namespace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the private key for BSF can be found	Mandatory	Not applicable

Table 3-29 (Cont.) Configurable Parameters for Ingress Gateway

Parameter	Description	Mandatory /Optional Parameter	Default Value
ingress-gateway.service.ssl.privateKey.rsa.fileName	File name for BSF's private key generated using the RSA algorithm	Mandatory	Not applicable
ingress-gateway.service.ssl.certificate.k8SecretName	Name of the Kubernetes Secret which contains the BSF Certificate.	Mandatory	Not applicable
ingress-gateway.service.ssl.certificate.k8Namespace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the BSF Certificate can be found.	Mandatory	Not Applicable
ingress-gateway.service.ssl.certificate.rsa.fileName	File name for BSF's Certificate, generated using an RSA resources.	Mandatory	Not Applicable
ingress-gateway.service.ssl.caBundle.k8SecretName	Name of the Kubernetes Secret which contains the Trust Chain Certificate.	Mandatory	Not Applicable
ingress-gateway.service.ssl.caBundle.k8Namespace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Trust Chain Certificate can be found.	Mandatory	Not Applicable
ingress-gateway.service.ssl.caBundle.fileName	File name for the Trust Chain Certificate	Mandatory	Not Applicable
ingress-gateway.service.ssl.keyStorePassword.k8SecretName	Name of the Kubernetes Secret which contains the Key Store Password file	Mandatory	Not Applicable
ingress-gateway.service.ssl.keyStorePassword.k8Namespace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Key Store Password file can be found.	Mandatory	Not Applicable
ingress-gateway.service.ssl.keyStorePassword.fileName	File name that has password for keyStore	Mandatory	Not Applicable
ingress-gateway.service.ssl.trustStorePassword.k8SecretName	Name of the Kubernetes Secret which contains the Trust Store Password file.	Mandatory	Not Applicable
ingress-gateway.service.ssl.trustStorePassword.k8Namespace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Trust Store Password file can be found.	Mandatory	Not Applicable
ingress-gateway.service.ssl.trustStorePassword.fileName	File name that has password for TrustStore	Mandatory	Not Applicable
ingressgateway.clientDisabledExtension	Disables the extension sent by messages originated by clients (ClientHello).	Optional	ec_point_formats

Table 3-29 (Cont.) Configurable Parameters for Ingress Gateway

Parameter	Description	Mandatory /Optional Parameter	Default Value
ingressgateway.serverDisabled Extension	Disables the extension sent by messages originated by servers (ServerHello).	Optional	null
ingressgateway.tlsNamedGroups	Provides a list of values sent in the supported_groups extension. These are comma-separated values.	Optional	null
ingressgateway.clientSignature Schemes	Provides a list of values sent in the signature_algorithms extension. These are comma-separated values.	Optional	null

Here is a sample configuration for configurable parameters in ocbf_custom_values_26.1.201.yaml file:

ingress-gateway:

```
#keep alive settings
ingressServer:
  keepAlive:
    enabled: false
    idealTime: 180 #in seconds
    count: 9
    interval: 60 #in seconds

allowedCipherSuites:
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

cipherSuites:
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

# ---- User Agent Validation Configuration - BEGIN ----
userAgentHeaderValidationConfigMode: HELM
userAgentHeaderValidation:
  enabled: false
  validationType: relaxed
  # List of consumer NF Types to be matched against the value of User-Agent
  header in the request
  consumerNfTypes:
    - "PCF"
    - "AF"
```

```

- "NEF"
# ---- User Agent Validation Configuration - END ----

# Enable it to accept incoming http requests
enableIncomingHttp: true

# ---- HTTPS Configuration - BEGIN ----
enableIncomingHttps: false

service:
  ssl:
    tlsVersion: TLSv1.2
    #supportedCipherSuiteList: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: ocegress.cer
    caBundle:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: caroot.cer
    keyStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: key.txt
    trustStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: trust.txt

# Enable or disable IP Address allocation from Metallb Pool
metallbIpAllocationEnabled: false

# Address Pool Annotation for Metallb
metallbIpAllocationAnnotation: "metallb.universe.tf/address-pool: signaling"
# -----Ingress Gateway Settings - END-----

```

3.14 Configuring Egress Gateway

This section describes the configuration parameters that are required for configurations in Egress Gateway.

Note

Following configurations are applicable only when Egress Gateway is enabled.

Table 3-30 Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.enableForwardedHeader	Enabling this parameter, egress-gateway will add Forwarded and x-Forwardedheaders. By default, the value is set to false.
egress-gateway.minAvailable	Specifies the number of pods that must always be available, even during a disruption.
egress-gateway.minReplicas	Specifies the minimum replicas to scale to maintain an average CPU utilization.
egress-gateway.maxReplicas	Specifies the maximum replicas to scale to maintain an average CPU utilization.
egress-gateway.userAgentHeaderConfigMode	This parameter is used to accept the user-agent configurations from Helm or REST.
egress-gateway.userAgentHeader.enabled	Specifies whether the feature is enabled or disabled. By default, the value is set to false.
egress-gateway.userAgentHeader.nfType	This parameter holds the nfType that will be used to generate the user agent header.
egress-gateway.userAgentHeader.nfInstanceId	This parameter represents the UUID of the CNPCF deployment that will be used to generate the user agent header.
egress-gateway.userAgentHeader.addFqdnToHeader	This parameter specifies if the user agent will use the FQDN information under the module to append it when generating the user agent header. The default value is set to 'false' meaning that the FQDN information will not be encoded into the user agent header during its generation. By default, the value is set to false.
egress-gateway.userAgentHeader.nfFqdn	This is an optional parameter and can be present or not, if operators want to include the FQDN string configured under this section then the parameter userAgentHeader.addFqdnToHeader needs to be enabled.
egress-gateway.userAgentHeader.overrideHeader	This parameter specifies if the the user agent header is sent or not.
egress-gateway.sniHeader.enabled	Enabling this parameter, egress-gateway will add SNI flag in client hello message of outbound traffic. Note: SNI enabling is depending on the initssl parameter from egress-gateway helm charts (Default value of initssl=true[TLS enable] , initssl=false[TLS disable]). It is an optional parameter. By default, the value is set to false.
egress-gateway.enableOutgoingHttps	This parameter is used to enable HTTPS for egress traffic. Default value: false

Table 3-30 (Cont.) Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.egressGwCertReloadEnabled	Default value: false
egress-gateway.egressGwCertReloadPath	Accepts a valid reload path. Default value: /egress-gw/store/reload
egress-gateway.service.ssl.privateKey.k8SecretName	Name of the Kubernetes Secret which contains the private key for BSF. Default value: Not applicable
egress-gateway.service.ssl.privateKey.k8NameSpace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the private key for BSF can be found Default value: Not applicable
egress-gateway.service.ssl.privateKey.rsa.fileName	File name for BSF's private key generated using the RSA algorithm Default value: Not applicable
egress-gateway.service.ssl.privateKey.ecdsa.fileName	File name for BSF's private key generated using the ECDSA algorithm Default value: Not applicable
egress-gateway.service.ssl.certificate.k8SecretName	Name of the Kubernetes Secret which contains the BSF Certificate. Default value: Not applicable
egress-gateway.service.ssl.certificate.k8NameSpace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the BSF Certificate can be found. Default value: Not applicable
egress-gateway.service.ssl.certificate.rsa.fileName	File name for BSF's Certificate, generated using an RSA resources. Default value: Not applicable
egress-gateway.service.ssl.certificate.ecdsa.fileName	File name for BSF's Certificate, generated using an ECDSA resources. Default value: Not applicable
egress-gateway.service.ssl.caBundle.k8SecretName	Name of the Kubernetes Secret which contains the Trust Chain Certificate. Default value: Not applicable
egress-gateway.service.ssl.caBundle.k8NameSpace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Trust Chain Certificate can be found. Default value: Not applicable
egress-gateway.service.ssl.caBundle.fileName	File name for the Trust Chain Certificate Default value: Not applicable
egress-gateway.service.ssl.keyStorePassword.k8SecretName	Name of the Kubernetes Secret which contains the Key Store Password file. Default value: Not applicable
egress-gateway.service.ssl.keyStorePassword.k8NameSpace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Key Store Password file can be found. Default value: Not applicable

Table 3-30 (Cont.) Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.service.ssl.keyStorePassword.fileName	File name that has password for keyStore Default value: Not applicable
egress-gateway.service.ssl.trustStorePassword.k8SecretName	Name of the Kubernetes Secret which contains the Trust Store Password file. Default value: Not applicable
egress-gateway.service.ssl.trustStorePassword.k8NameSpace	Name of the Kubernetes Namespace where the Kubernetes Secret containing the Trust Store Password file can be found. Default value: Not applicable
egress-gateway.service.ssl.trustStorePassword.fileName	File name that has password for TrustStore. Default value: Not applicable
egress-gateway.service.ssl.tlsVersion	Indicates the TLS version, a mandatory field. Data Type: String Default Value: TLSv1.2 Range: • TLSv1.2 • TLSv1.3
egress-gateway.allowedCipherSuites	Indicates the allowed Ciphers suites. Data Type: String Default Value: NA Range: • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Table 3-30 (Cont.) Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.cipherSuites	Indicates the supported cipher suites. Data Type: String Default Value: NA Range: • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_AES_128_GCM_SHA256 • TLS_CHACHA20_POLY1305_SHA256
egress-gateway.message-copy.enabled	Indicates whether to enable or disable message copy at the Gateway. Default value: false
egress-gateway.message-copy.copyPayload	Indicates whether to enable or disable message payload (HTTP message body) in the feed towards Oracle Communications Network Analytics Data Director (OCNADD). Default value: true
egress-gateway.message-copy.topicName	List of comma-separated Kafka Nodes.
egress-gateway.message-copy.ackRequired	Indicates whether to whether to wait for acknowledgement from Kafka or not. Default value: false
egress-gateway.message-copy.retryOnFailure	Specifies the number of times Egress Gateway must retry if the message was not sent to Kafka successfully. Default value: 0
egress-gateway.message-copy.threadPoolConfigurations.coreSize	Specifies the core size of the thread pool. Default value: 8

Table 3-30 (Cont.) Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.message-copy.threadPoolConfigurations.maxSize	Specifies the maximum size of the thread pool. Default value: 8 Note Configuring threadPoolConfigurations.coreSize and threadPoolConfigurations.maxSize ensures the fixed size thread pool, which eliminates the performance overhead of thread creation at runtime.
egress-gateway.message-copy.threadPoolConfigurations.queueCapacity	Specifies the the maximum number of message copy events that may remain on a queue once published. Default value: 1000
egress-gateway.message-copy.security.enabled	Enables the SSL/SASL_SSL based communication between NRF and OCNADD. Default value: false
egress-gateway.message-copy.security.protocol	This parameter is used to define the security mechanism using which NRF and OCNADD will communicate over the wire. Default value: SASL_SSL
egress-gateway.message-copy.security.tlsVersion	This parameter is used to define the supported TLS version by NRF, which will be used during TLS version negotiation. Default value: TLSv1.2
egress-gateway.message-copy.security.saslConfiguration.username	This parameters is used to define the username that will be used by NRF to authenticate itself with DD if the messageCopy.security.protocol=SASL_SSL.

Table 3-30 (Cont.) Configurable Parameters for Configurations in Egress Gateway

Parameter	Description
egress-gateway.message-copy.security.saslConfiguration.username.password.k8SecretName	This parameters is used to define the password that will be used by NRF to authenticate itself with DD if the messageCopy.security.protocol=SASL_SSL. The password is stored in a k8s secret. security.saslConfiguration.username.password.k8SecretName stores the secret name.
egress-gateway.message-copy.security.saslConfiguration.username.password.k8Namespace	Stores the name of the NRF deployment namespace.
egress-gateway.message-copy.security.saslConfiguration.username.password.fileName	Indicates the password file which is used to create SSL secret.
egress-gateway.kafka.bootstrapAddress	Indicates the bootstrap address of the broker from where the Kafka client can retrieve the metadata of the clusters.

Here is a sample configuration for configurable parameters in ocbsf_custom_values_26.1.201.yaml file:

egress-gateway:

```
# enabling this egress-gateway will add Forwarded and x-Forwardedheaders
enableForwardedHeader: false

#Cipher Suites to be enabled on client side
cipherSuites:
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

allowedCipherSuites:
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

# ---- User-agent Header configuration - BEGIN ----
userAgentHeaderConfigMode: HELM
userAgentHeader:
  enabled: false # flag to enable or disable the feature
  nfType: "" # NF type of consumer NF
  nfInstanceId: "" # NF type of consumer NF
  addFqdnToHeader: false # Flag to add fqdn. If enabled then user-agent
header will be generated along with the fqdn configured otherwise fqdn will
```

```

not be added
  nfFqdn: "" #fqdn of NF. This is not the fqdn of gateway
  overwriteHeader: false
# ---- User-agent Header Configuration - END ----

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

#Enabling it for egress https requests
enableOutgoingHttps: false

#Enabling it for egress http1.1 requests
http1:
  enableOutgoingHTTP1: false # Flag to enable or disable the feature

egressGwCertReloadEnabled: false
egressGwCertReloadPath: /egress-gw/store/reload

service:
  ssl:
    tlsVersion: TLSv1.2
    #supportedCipherSuiteList: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: rsa_private_key_pkcs1.pem
      ecdsa:
        fileName: ssl_ecdsa_private_key.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      rsa:
        fileName: ocegress.cer
      ecdsa:
        fileName: ssl_ecdsa_certificate.crt
    caBundle:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: caroot.cer
    keyStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: key.txt
    trustStorePassword:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
      fileName: trust.txt
# ---- HTTPS Configuration - END ----
message-copy:
  enabled: false
  copyPayload: false
  topicName: bsfMessageFeed
  ackRequired: false
  retryOnFailure: 0
  security:
    enabled: false

```

```

protocol: SASL_SSL
tlsVersion: TLSv1.2
saslConfiguration:
  userName: test
  password:
    k8SecretName: message-copy-secret
    k8Namespace: bsf
    fileName: password.txt
threadPoolConfigurations:
  coreSize: 8
  maxSize: 8
  queueCapacity: 1000

kafka:
  bootstrapAddress:

```

3.15 Configuring Service and Container Ports

This section describes the customizations that you can make in `ocbsf_custom_values_26.1.201.yaml` file to configure service and container ports.

Note

For upgrade scenario, changing port will cause temporary service disruption.

To override the default port numbers, used by service and container ports, and customize them as per your requirements, you can configure the following configurable parameters in `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-31 Customizable Parameters for Service Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
<code>global.servicePorts.bsfManagementServiceHttp</code>	HTTP signaling port for BSF management service.	Optional	5903
<code>global.servicePorts.bsfManagementServiceHttps</code>	HTTPS signaling port for BSF management service.	Optional	8443
<code>global.servicePorts.appInfoHttp</code>	HTTP signaling port for app info. Note: The value for this port must be same as <code>svcAppInfoHttp</code>	Optional	5906
<code>global.servicePorts.cmServiceHttp</code>	HTTP signaling port for CM service.	Optional	5808
<code>global.servicePorts.configServerHttp</code>	HTTP signaling port for config server. Note: The value for this port must be same as <code>svcConfigServerHttp</code>	Optional	5807
<code>global.servicePorts.diamGatewayHttp</code>	HTTP signaling port for Diameter gateway.	Optional	8080

Table 3-31 (Cont.) Customizable Parameters for Service Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.servicePorts.diamGatewayDiameter	Port for Diameter gateway.	Optional	3868
global.servicePorts.perfInfoHttp	HTTP signaling port for perf info. The value for this port must be same as svcPerfInfoHttp.	Optional	5905
global.servicePorts.queryServiceHttp	HTTP signaling port for query service.	Optional	5805
global.servicePorts.auditServiceHttp	This HTTP signaling port is used for audit service.	Optional	8000
global.servicePorts.egressGatewayHttp	HTTP signaling port for Egress Gateway. The value for this port must be same as svcEgressGatewayHttp.	Optional	8080
global.servicePorts.nrfClientNfManagementHttp	HTTP signaling port for NRF client management service. The value for this port must be same as svcNrfClientNfManagementHttp.	Optional	5910
global.servicePorts.nrfClientNfManagementHttps	HTTPS signaling port for NRF client management service. The value for this port must be same as svcNrfClientNfManagementHttps.	Optional	5805
global.servicePorts.nrfClientNfDiscoveryHttp	HTTP signaling port for NF discovery service by NRF client. The value for this port must be same as svcNrfClientNfDiscoveryHttp.	Optional	8000
global.servicePorts.nrfClientNfDiscoveryHttps	HTTP signaling port for NF discovery service by NRF client. The value for this port must be same as svcNrfClientNfDiscoveryHttps.	Optional	9443
global.servicePorts.alternateRouteServiceHttp	HTTP signaling port for alternate route service. The value for this port must be same as svcAlternateRouteServiceHttp.	Optional	8000

Table 3-31 (Cont.) Customizable Parameters for Service Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.servicePorts.alternateRouteServiceHazelcast	The value for this port must be same as svcAlternateRouteServiceHazelcast.	Optional	8000

Here is a sample of service ports configurable parameters in ocbsf_custom_values_26.1.201.yaml file:

```

servicePorts:
  bsfManagementServiceHttp: 5903
  bsfManagementServiceHttps: 8443
  # app info
  appInfoHttp: &svcAppInfoHttp 8000
  # cm service
  cmServiceHttp: &svcCmServiceHttp 8000
  # config server
  configServerHttp: &svcConfigServerHttp 8000
  # diameter gateway
  diamGatewayHttp: 8000
  diamGatewayDiameter: 3868
  # perf info
  perfInfoHttp: &svcPerfInfoHttp 8000
  # query service
  queryServiceHttp: 8000
  # audit service
  auditServiceHttp: 8000
  # egress gateway
  egressGatewayHttp: &svcEgressGatewayHttp 8000
  # nrf client
  nrfClientNfDiscoveryHttp: &svcNrfClientNfDiscoveryHttp 8000
  nrfClientNfManagementHttp: &svcNrfClientNfManagementHttp 8000
  nrfClientNfDiscoveryHttps: &svcNrfClientNfDiscoveryHttps 9443
  nrfClientNfManagementHttps: &svcNrfClientNfManagementHttps 9443
  # alternate route
  alternateRouteServiceHttp: &svcAlternateRouteServiceHttp 8000
  alternateRouteServiceHazelcast: &svcAlternateRouteServiceHazelcast 8000

```

Table 3-32 Customizable Parameters for Container Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.containerPorts.monitoringHttp	HTTP signaling port for monitoring. Note: The value for this port must be same as containerMonitoringHttp.	Optional	9000

Table 3-32 (Cont.) Customizable Parameters for Container Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.containerPorts.bsfManagementServiceHttp	HTTP signaling port for BSF Management service.	Optional	8080
global.containerPorts.bsfManagementServiceHttps	HTTPS signaling port for BSF Management service.	Optional	8443
global.containerPorts.appInfoHttp	HTTP signaling port for app info.	Optional	5906
global.containerPorts.cmServiceHttp	HTTP signaling port for CMservice.	Optional	5807
global.containerPorts.configServerHttp	HTTP signaling port for config server.	Optional	8001
global.containerPorts.diamGatewayHttp	HTTP signaling port for Diameter Gateway.	Optional	8080
global.containerPorts.diamGatewayDiameter	Diameter gateway.	Optional	3868
global.containerPorts.perfInfoHttp	HTTP signaling port for perf-info.	Optional	5905
global.containerPorts.queryServiceHttp	HTTP signaling port for queryservice.	Optional	8081
global.containerPorts.auditServiceHttp	HTTP signaling port for audit service.	Optional	8000
global.containerPorts.nrfClientNfManagementHttp	HTTP signaling port for NRF client management. Note: The value for this port must be same as containerNrfClientNfManagementHttp.	Optional	8000
global.containerPorts.nrfClientNfDiscoveryHttp	HTTP signaling port for NF discovery service by NRF client. The value for this port must be same as containerNrfClientNfDiscoveryHttp.	Optional	8000
global.containerPorts.nrfClientNfManagementHttps	HTTPS signaling port for NRF client management. Note: The value for this port must be same as containerNrfClientNfManagementHttps.	Optional	9443
global.containerPorts.nrfClientNfDiscoveryHttps	HTTPS signaling port for NF discovery service by NRF client. The value for this port must be same as containerNrfClientNfDiscoveryHttps.	Optional	9443

Table 3-32 (Cont.) Customizable Parameters for Container Ports Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.containerPorts.ingressGatewayHttp	HTTP signaling port for Ingress Gateway. Note: The value for this port must be same as containerIngressGatewayHttp.	Optional	8081
global.containerPorts.ingressGatewayHttps	HTTPS signaling port for Ingress Gateway. Note: The value for this port must be same as containerIngressGatewayHttps.	Optional	9443
global.containerPorts.alternateRouteServiceHttp	HTTP Signaling port for alternate route service. Note: The value for this port must be same as containerAlternateRouteServiceHttp.	Optional	8004

Here is a sample of service ports configurable parameters in ocbf_custom_values_26.1.201.yaml file:

```

containerPorts:
  bsfManagementServiceHttp: 8080
  bsfManagementServiceHttps: 8443
  monitoringHttp: &containerMonitoringHttp 9000
  # app info
  appInfoHttp: 8000
  # cm service
  cmServiceHttp: 8000
  # config server
  configServerHttp: 8000
  # diameter gateway
  diamGatewayHttp: 8000
  diamGatewayDiameter: 3868
  # perf info
  perfInfoHttp: 8000
  # query service
  queryServiceHttp: 8000
  # audit service
  auditServiceHttp: 8000
  # nrf client
  nrfClientNfDiscoveryHttp: &containerNrfClientNfDiscoveryHttp 8000
  nrfClientNfManagementHttp: &containerNrfClientNfManagementHttp 8000
  nrfClientNfDiscoveryHttps: &containerNrfClientNfDiscoveryHttps 9443
  nrfClientNfManagementHttps: &containerNrfClientNfManagementHttps 9443
  # ingress gateway
  ingressGatewayHttp: &containerIngressGatewayHttp 8000
  ingressGatewayHttps: &containerIngressGatewayHttps 9443

```

alternate route service : Note: This port shall not be same as
alternateRouteServiceHazelcast which is 8000 in this sample custom values file
alternateRouteServiceHttp: &containerAlternateRouteServiceHttp 8004

Table 3-33 Customizable Parameters for Ports Configuration in Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
global.publicHttpSignalingPort	HTTP/2.0 Port of ingress gateway	Optional	80
global.publicHttpsSignalingPort	HTTPS/2.0 Port of ingress gateway The value for this port must be set to 0 if HTTPS is disabled.	Optional	443
global.configServerPort	HTTP signaling port for config server.	Optional	Note: The value for this port must be same as svcConfigServerHttp.
ingress-gateway.ports.actuatorPort		Optional	Same value as containerMonitoringHttp
ingress-gateway.ports.containerPort		Optional	Same value as containerIngressGatewayHttp
ingress-gateway.ports.containersslPort		Optional	Same value as containerIngressGatewayHttps

Here is a sample of configurable parameters for ingress-gateway's ports in ocbsf_custom_values_26.1.201.yaml file:

```
# -----Ingress Gateway Settings - BEGIN-----
# If httpsEnabled is false, this Port would be HTTP/2.0 Port (unsecured)
publicHttpSignalingPort: 80
# If httpsEnabled is true, this Port would be HTTPS/2.0 Port (secured SSL)
publicHttpsSignalingPort: 443

configServerPort: *svcConfigServerHttp

ingress-gateway:
  ports:
    actuatorPort: *containerMonitoringHttp
    containerPort: *containerIngressGatewayHttp
    containersslPort: *containerIngressGatewayHttps
```

Table 3-34 Customizable Parameters for Ports Configuration in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
egress-gateway.serviceEgressGateway.actuatorPort		Optional	Same value as containerMonitoringHttp
egress-gateway.serviceEgressGateway.Port		Optional	Same value as svcEgressGatewayHttp

Here is a sample of configurable parameters for egress-gateway's ports in ocbssf_custom_values_26.1.201.yaml file:

```
egress-gateway:
  serviceEgressGateway:
    actuatorPort: *containerMonitoringHttp
    port: *svcEgressGatewayHttp
```

Table 3-35 Customizable Parameters for Ports Configuration in nrf-client-nfdiscovery

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
global.nrf-client-nfdiscovery.envPlatformServicePort	HTTP signaling port for app info.	Optional	5906	Same value as svcAppInfoHttp
global.nrf-client-nfdiscovery.envPerformanceServicePort	HTTP signaling port for perf info.	Optional	5905	Same value as svcPerfInfoHttp
global.nrf-client-nfdiscovery.envCfgServerPort	HTTP signaling port for config server.	No	5807	same vale as svcConfigServerHttp
global.nrf-client-nfdiscovery.containerHttpPort	HTTP signaling port for NRF client discovery.	Optional	8000	Same value as containerNrfClientNfDiscoveryHttp
global.nrf-client-nfdiscovery.containerHttpsPort	HTTPS signaling port for NRF client discovery.	Optional	9443	Same value as containerNrfClientNfDiscoveryHttps

Table 3-35 (Cont.) Customizable Parameters for Ports Configuration in nrf-client-nfdiscovery

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
global.nrf-client-nfdiscovery.serviceHttpPort	HTTP signaling port for NRF client discovery service.	Optional	5910	Same value as svcNrfClientNfDiscoveryHttp
global.nrf-client-nfdiscovery.serviceHttpsPort	HTTPS signaling port for NRF client discovery service.	Optional	8443	Same value as svcNrfClientNfDiscoveryHttps

Here is a sample of configurable parameters for nrf-client-nfdiscovery's ports in ocbf_custom_values_26.1.201.yaml file:

```
nrf-client-nfdiscovery:
  envJaegerSamplerParam: '1'
  envJaegerSamplerType: ratelimiting
  envJaegerServiceName: pcf-nrf-client-nfdiscovery
  envPlatformServicePort: *svcAppInfoHttp
  envPerformanceServicePort: *svcPerfInfoHttp
  envCfgServerPort: *svcConfigServerHttp
  containerHttpPort: *containerNrfClientNfDiscoveryHttp
  containerHttpsPort: *containerNrfClientNfDiscoveryHttps
  serviceHttpPort: *svcNrfClientNfDiscoveryHttp
  serviceHttpsPort: *svcNrfClientNfDiscoveryHttps
  envDiscoveryServicePort: *svcNrfClientNfDiscoveryHttp
  envManagementServicePort: *svcNrfClientNfManagementHttp
  alternateRouteServiceEnabled: false
```

Table 3-36 Customizable Parameters for Ports Configuration in nrf-client-nfmanagement

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Value
global.nrf-client-nfmanagement.envPlatformServicePort	HTTP signaling port for app info.	Optional	5906	Same value as svcAppInfoHttp
global.nrf-client-nfmanagement.envPerformanceServicePort	HTTP signaling port for perf info.	Optional	5905	Same value as svcPerfInfoHttp
global.nrf-client-nfmanagement.envCfgServerPort	HTTP signaling port for config server.	Optional	5807	same vale as svcConfigServerHttp

Table 3-36 (Cont.) Customizable Parameters for Ports Configuration in nrf-client-nfmanagement

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Value
global.nrf-client-nfmanagement.containerHttpPort	HTTP signaling port for NRF client discovery.	Optional	8000	Same value as containerNrfClientNfManagementHttp
global.nrf-client-nfmanagement.containerHttpsPort	HTTPS signaling port for NRF client discovery.	Optional	9443	Same value as containerNrfClientNfManagementHttps
global.nrf-client-nfmanagement.serviceHttpPort	HTTP signaling port for NRF client discovery service.	Optional	5910	Same value as svcNrfClientNfManagementHttp
global.nrf-client-nfmanagement.serviceHttpsPort	HTTPS signaling port for NRF client discovery service.	Optional	8443	Same value as svcNrfClientNfManagementHttps

Here is a sample of configurable parameters for nrf-client-nfmanagement's ports in ocbsf_custom_values_26.1.201.yaml file:

```
nrf-client-nfmanagement:
  envJaegerSamplerParam: '1'
  envJaegerSamplerType: ratelimiting
  envJaegerServiceName: pcf-nrf-client-nfmanagement
  envPlatformServicePort: *svcAppInfoHttp
  envPerformanceServicePort: *svcPerfInfoHttp
  envCfgServerPort: *svcConfigServerHttp
  containerHttpPort: *containerNrfClientNfManagementHttp
  containerHttpsPort: *containerNrfClientNfManagementHttps
  serviceHttpPort: *svcNrfClientNfManagementHttp
  serviceHttpsPort: *svcNrfClientNfManagementHttps
  alternateRouteServiceEnabled: false
```

Table 3-37 Customizable Parameters for Ports Configuration in Alternate Route Service

Parameter	Description	Mandatory/Optional Parameter	Default Value	Notes
alternate-route.ports.servicePort	HTTP signaling port for alternate route service.	Optional	8000	Same value as svcAlternateRouteServiceHttp
alternate-route.ports.containerPort	HTTP signaling port for alternate route service.	Optional	8004	Same value as containerAlternateRouteServiceHttp
alternate-route.ports.actuatorPort	HTTP signaling port for monitoring.	Optional	9000	Same value as containerMonitoringHttp
alternate-route.hazelcast.port		Optional	8000	Same value as svcAlternateRouteServiceHazelcast

Here is a sample of configurable parameters for alternate route service's ports in ocbsf_custom_values_26.1.201.yaml file:

```
alternate-route:
  ports:
    servicePort: *svcAlternateRouteServiceHttp
    containerPort: *containerAlternateRouteServiceHttp
    actuatorPort: *containerMonitoringHttp
  hazelcast:
    port: *svcAlternateRouteServiceHazelcast
```

3.16 OAUTH Configuration

This section describes the customizations that you should make in ocbsf_custom_values_26.1.201.yaml files to configure OAUTH in ingress/egress gateway.

Note

These configurations are applicable when the Ingress/Egress Gateway is enabled and the NRF Client services are enabled.

To configure OAUTH in ingress-gateway, you should configure the following configurable parameters in custom-value.yaml file:

Table 3-38 Configurable Parameters for OAUTH Configuration in Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
ingress-gateway.oauthValidatorEnabled	Enable or disable OAuth Validator. If Access Token service is not required, operator can choose to set the value of this parameter as false. By doing so, Access Token microservice will not be deployed.	Optional	false
ingress-gateway.nfInstanceId	NF Instance Id of service producer	Optional	6faf1bbc-6e4a-4454-a507-a14ef8e1bc11
ingress-gateway.allowedClockSkewSeconds	set this value if clock on the parsing NF (producer) is not perfectly in sync with the clock on the NF (consumer) that created by JWT	Optional	0
ingress-gateway.nrfPublicKeyKubeSecret	Name of the secret which stores the public key(s) of NRF	Optional	
ingress-gateway.nrfPublicKeyKubeNamespace	Namespace of the NRF public key secret	Optional	
ingress-gateway.validationType	Possible values are: - strict - relaxed strict- If incoming request does not contain "Authorization" (Access Token) header, the request is rejected. relaxed- relaxed means that if Incoming request contains "Authorization" header, it is validated. If Incoming request does not contain "Authorization" header, validation is ignored.	Optional	relaxed
ingress-gateway.producerPlmnMNC	MNC of the service producer	Optional	123
ingress-gateway.producerPlmnMCC	MCC of the service producer	Optional	456
ingress-gateway.producerScope	Contains the NF service name(s) of the NF service producer(s). The service name(s) included in this attribute shall be any of the services defined in the ServiceName enumerated type. Note: producerScope must be configured in custom-values.yaml only if different from the default values.	Mandatory	nbsf-management

Table 3-38 (Cont.) Configurable Parameters for OAUTH Configuration in Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
ingress-gateway.nfType	Specifies the NF type of the NF service producer. It is included when the access token request is for an NF or NF service instance.	Optional	BSF

Here is a sample OAUTH configurations in ingress-gateway in ocbf_custom_values_26.1.201.yaml file:

```
# ----OAUTH CONFIGURATION - BEGIN ----
oauthValidatorEnabled: false
nfInstanceId: 6faf1bbc-6e4a-4454-a507-a14ef8e1bc11
allowedClockSkewSeconds: 0
nrfPublicKeyKubeSecret: ''
nrfPublicKeyKubeNamespace: ''
validationType: relaxed
producerPlmnMNC: 123
producerPlmnMCC: 456
nfType: BSF
# ----OAUTH CONFIGURATION - END ----
```

Table 3-39 Configurable Parameters for OAUTH Configuration in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
egress-gateway.oauthClient.enabled	OAuth Validator Enabled	Optional	false
egress-gateway.oauthClient.dnsSrvEnabled	Enable/Dsiable the DNS-SRV query to coreDNS Server	Optional	false
egress-gateway.oauthClient.httpsEnabled	Determine if https support is enabled or not which is a deciding factor for oauth request scheme and search query parameter in dns-srv request.	Optional	false
egress-gateway.oauthClient.nrfClientQueryEnabled	Determines if NRF-Client Query is enabled or not (Dynamic configuration).	Optional	false
egress-gateway.oauthClient.virtualFqdn	Indicates the name of the virtual FQDN or FQDN that needs to be sent in the dns-srv query.	Conditional (If dnsSrvEnabled is set to true.)	string Example: nrf.oracle.com:80
egress-gateway.oauthClient.staticNrfList	List of Static NRF's	Conditional (If oAuth is enabled.)	

Table 3-39 (Cont.) Configurable Parameters for OAUTH Configuration in Egress Gateway

Parameter	Description	Mandatory/Optional Parameter	Default Value
egress-gateway.oauthClient.nfInstanceId	NF InstanceId of Producer	Optional	fe7d992b-0541-4c7d-ab84-c6d70b1b01b1 Note: Update the parameter with actual value, if OAuth is enabled.
egress-gateway.oauthClient.consumerPImnMNC	MNC of service Consumer	Optional	345 Note: Update the parameter with actual value, if OAuth is enabled.
egress-gateway.oauthClient.consumerPImnMCC	MCC of service Consumer	Optional	567 Note: Update the parameter with actual value, if OAuth is enabled.
egress-gateway.oauthClient.maxRetry	Maximum number of retry that need to be performed to other NRF Fqdn's in case of failure response from first contacted NRF based on the errorCodeSeries configured.	Conditional (If OAuth is enabled.)	2
egress-gateway.oauthClient.apiPrefix	apiPrefix that needs to be appended in the Oauth request flow.	Conditional (If OAuth is enabled.)	
egress-gateway.oauthClient.errorCodeSeries	Determines the fallback condition to other NRF in case of failure response from currently contacted NRF.	Conditional (If OAuth is enabled and required a different error code series.)	4XX
egress-gateway.oauthClient.retryAfter	RetryAfter value in milliseconds that needs to be set for a particular NRF Fqdn, if the error matched the configured errorCodeSeries.	Conditional (If OAuth is enabled.)	5000
egress-gateway.oauthClient.nrfClientConfig	Determines the NRF-Client Mgmt Svc configurations which are required when dynamic configurations are in place at Egress-Gateway.		
egress-gateway.oauthClient.nrfClientConfig.serviceName	The service name of NRF-Client Mgmt Svc		ocbsf-nrf-client-nfmanagement

Table 3-39 (Cont.) Configurable Parameters for OAUTH Configuration in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
egress-gateway.oauthClient.nrfClientConfig.host	The address of NRF-Client Mgmt Svc		10.233.49.44
egress-gateway.oauthClient.nrfClientConfig.port	Determines the port configuration for NRF-Client Mgmt Svc for sending Subscription requests.		8000
egress-gateway.oauthClient.nrfClientRequestMap	Determines the request mapping URL for sending Subscription requests from Egress-Gateway to NRF-Client Mgmt Svc.		/v1/nrf-client/subscriptions/nrfRouteList

Here is a sample OAUTH configurations in egress-gateway in ocbsf_custom_values_26.1.201.yaml file:

```
oauthClient:
  enabled: false
  dnsSrvEnabled: false
  httpsEnabled: false
  nrfClientQueryEnabled: false
  virtualFqdn: nrf.oracle.com:80
  staticNrfList:
    - nrf1.oracle.com:80
  nfInstanceId: fe7d992b-0541-4c7d-ab84-c6d70b1b01b1
  consumerPlmnMNC: 345
  consumerPlmnMCC: 567
  maxRetry: 2
  apiPrefix: ""
  errorCodeSeries: 4XX
  retryAfter: 5000
  nrfClientConfig:
    serviceName: "ocbsf-nrf-client-nfmanagement"
    host: 10.233.49.44
    port: 8000
    nrfClientRequestMap: "/v1/nrf-client/subscriptions/nrfRouteList"
# ---- Oauth Configuration - END ----
```

3.17 Configuring Ingress/Egress Gateway HTTPS

This section describes the customizations that you should make in ocbsf_custom_values_26.1.201.yaml files to configure HTTPS in ingress/egress gateway.

Note

These configurations are applicable only when ingress/egress gateway is enabled and the following parameters are set to true in `ocbsf_custom_values_26.1.201.yaml` file:

- `ingress-gateway.enableIncomingHttps`
- `egress-gateway.enableOutgoingHttps`

To configure HTTPS in ingress-gateway, you should configure the following configurable parameters in `custom-value.yaml` file:

Table 3-40 Configurable Parameters for HTTPS Configurations in Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
<code>ingress-gateway.enableIncomingHttps</code>	To enable https for ingress traffic	Optional	False	
<code>ingress-gateway.service.ssl.privateKey.k8SecretName</code>	Name of the private key secret.	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.privateKey.k8Namespace</code>	Namespace of private key.	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.privateKey.rsa.fileName</code>	rsa private key file name.	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.certificate.k8SecretName</code>	Name of the private key secret	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.certificate.k8Namespace</code>	Namespace of private key	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.certificate.rsa.fileName</code>	rsa private key file name	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.caBundle.k8SecretName</code>	Name of the private key secret	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true
<code>ingress-gateway.service.ssl.caBundle.k8Namespace</code>	Namespace of private key	Optional	Not Applicable	required if <code>enableIncomingHttps</code> is true

Table 3-40 (Cont.) Configurable Parameters for HTTPS Configurations in Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
ingress-gateway.service.ssl.caBundle.fileName	private key file name	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.keyStorePassword.k8SecretName	Name of the privatekey secret	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.keyStorePassword.k8Namespace	Namespace of privatekey	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.keyStorePassword.fileName	File name that has password for keyStore	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.trustStorePassword.k8SecretName	Name of the privatekey secret	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.trustStorePassword.k8Namespace	Namespace of privatekey	Optional	Not Applicable	required if enableIncomingHttps is true
ingress-gateway.service.ssl.trustStorePassword.fileName	File name that has password for trustStore	Optional	Not Applicable	required if enableIncomingHttps is true

Here is a sample HTTPS configurations in ingress-gateway in ocbsf_custom_values_26.1.201.yaml file:

```
# ---- HTTPS Configuration - BEGIN ----
enableIncomingHttps: false

service:
  ssl:
    privateKey:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
    rsa:
      fileName: rsa_private_key_pkcs1.pem
    certificate:
      k8SecretName: ocbsf-gateway-secret
      k8Namespace: ocbsf
    rsa:
      fileName: ocegress.cer
  caBundle:
    k8SecretName: ocbsf-gateway-secret
    k8Namespace: ocbsf
```

```

fileName: caroot.cer
keyStorePassword:
k8SecretName: ocbsf-gateway-secret
k8Namespace: ocbsf
fileName: key.txt
trustStorePassword:
k8SecretName: ocbsf-gateway-secret
k8Namespace: ocbsf
fileName: trust.txt

```

Table 3-41 Configurable Parameters for HTTPS Configurations in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
egress-gateway.enableOutgoingHttps	Enabling it for outgoing https request	No	false	
egress-gateway.egressGwCertReloadEnabled		No	false	
egress-gateway.egressGwCertReloadPath		No	/egress-gw/store/reload	
egress-gateway.service.ssl.privateKey.k8SecretName	Name of the privatekey secret	No	Not Applicable	
egress-gateway.service.ssl.privateKey.k8Namespace	Namespace of privatekey	No	Not Applicable	
egress-gateway.service.ssl.privateKey.rsa.fileName	rsa private key file name	No	Not Applicable	
egress-gateway.service.ssl.privateKey.ecdsa.fileName	ecdsa private key file name	No	Not Applicable	
egress-gateway.service.ssl.certificate.k8SecretName	Name of the privatekey secret	No	Not Applicable	
egress-gateway.service.ssl.certificate.k8Namespace	Namespace of privatekey	No	Not Applicable	
egress-gateway.service.ssl.certificate.rsa.fileName	rsa private key file name	No	Not Applicable	
egress-gateway.service.ssl.certificate.ecdsa.fileName	ecdsa private key file name	No	Not Applicable	
egress-gateway.service.ssl.caBundle.k8SecretName	Name of the privatekey secret	No	Not Applicable	
egress-gateway.service.ssl.caBundle.k8Namespace	Namespace of privatekey	No	Not Applicable	

Table 3-41 (Cont.) Configurable Parameters for HTTPS Configurations in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
egress-gateway.service.ssl.caBundle.fileName	private key file name	No	Not Applicable	
egress-gateway.service.ssl.keyStorePassword.k8SecretName	Name of the privatekey secret	No	Not Applicable	
egress-gateway.service.ssl.keyStorePassword.k8Namespace	Namespace of privatekey	No	Not Applicable	
egress-gateway.service.ssl.keyStorePassword.fileName	File name that has password for keyStore	No	Not Applicable	
egress-gateway.service.ssl.trustStorePassword.k8SecretName	Name of the privatekey secret	No	Not Applicable	
egress-gateway.service.ssl.trustStorePassword.k8Namespace	Namespace of privatekey	No	Not Applicable	
egress-gateway.service.ssl.trustStorePassword.fileName	File name that has password for trustStore	No	Not Applicable	

Here is a sample HTTPS configurations in egress-gateway in ocbf_custom_values_26.1.201.yaml file:

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

#Enabling it for egress https requests
enableOutgoingHttps: false

egressGwCertReloadEnabled: false
egressGwCertReloadPath: /egress-gw/store/reload

service:
  ssl:
    privateKey:
      k8SecretName: ocbf-gateway-secret
      k8Namespace: ocbf
    rsa:
      fileName: rsa_private_key_pkcs1.pem
    ecDSA:
      fileName: ssl_ecdsa_private_key.pem
  certificate:
    k8SecretName: ocbf-gateway-secret
    k8Namespace: ocbf
    rsa:
      fileName: ocbf.egress.cer
    ecDSA:
```

```

        fileName: ssl_ecdsa_certificate.crt
    caBundle:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: caroot.cer
    keyStorePassword:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: key.txt
    trustStorePassword:
        k8SecretName: ocbsf-gateway-secret
        k8Namespace: ocbsf
        fileName: trust.txt
# ---- HTTPS Configuration - END ----

```

3.18 Configuring SCP

This section describes the customizations that you can make in `ocbsf_custom_values_26.1.201.yaml` files to support SCP integration including SBI routing.

! Important

- Routes supporting the SBI-Routing configuration are updated in Egress Gateway only when its configuration details are provided correctly. Example: `PeerSetConfiguration`, `PeerConfiguration`, `sbiroutingerrorcriteriasets`, and `sbiroutingerroractionsets`. Routes not supporting the SBI-Routing configuration are updated only when they have valid route definition.

To configure SBI-Routing:

- Use `Peerconfiguration` to define the list of peers to which Egress Gateway can send request. This list contains peers that support HTTP/ HTTP-Proxy / HTTPS communication.
- Use `Peersetconfiguration` to logically group the peers into sets. Each set contains a list of peers that support HTTP and HTTPS communication modes.
- Use `sbiRoutingErrorCriteriaSets` to define an array of `errorCriteriaSet` , where each `errorCriteriaSet` depicts an ID, set of HTTP Methods, set of HTTP Response status codes set of exceptions with `headerMatching` functionality.
- Use `sbiRoutingErrorActionSets` to define an array of `actionset`, where each depicts an ID, action to be performed (Currently on REROUTE action is supported) and blacklist configurations.
- Configure the `Priority` for each peer in the set. Depending on the priority, it selects the primary, secondary, or tertiary peers to route requests.

Note

- Egress Gateway accepts route configuration updates only if SBI-Routing feature is configured correctly.
- If the peer contains a virtual host address, Egress Gateway resolves the virtual host address using DNS-SRV query. If a peer is defined based on virtual host, then peerset can contain only one such peer for `httpconfiguration` and `httpsconfiguration`. User should not configure more than one virtual host based on peer in a given peerset for a given HTTP / HTTPS configuration.
- In case of peers based on virtual host, Egress Gateway does not consider priority values configured rather it retrieves priority from DNS-SRV records.

The following flags determine whether the configuration for routes and sbiRouting needs to be picked up from Helm

```
routeConfigMode: HELM
```

Configurations for SBI Routing

To enable and configure SBI Routing, perform the following configurations

- For `sbiRoutingDefaultScheme` parameter, the default value is `http`. The value specified in this field is considered when `3gpp-sbi-target-apiroot` header is missing.
- Now, configure a list of peers and peer sets. Each peer must contain `id`, `host`, `port`, and `apiPrefix`. Each peer set must contain HTTP or HTTPS instances where in each instance contains priority and peer identifier, which maps to peers configured under `peerConfiguration`.
No two instances should have same priority for a given HTTP or HTTPS configuration. In addition, more than one virtual FQDN should not be configured for a given HTTP or HTTPS configuration.

```
sbiRouting:
```

```
# Default scheme applicable when 3gpp-sbi-target-apiroot header is missing
sbiRoutingDefaultScheme: http
```

```
peerConfiguration:
```

- ```
- id: peer1
 host: scp1.test.com
 port: 80
 apiPrefix: "/"
- id: peer2
 host: scp2.test.com
 port: 80
 apiPrefix: "/"
```

```
peerSetConfiguration:
```

- ```
- id: set0
  httpConfiguration:
    - priority: 1
      peerIdentifier: peer1
    - priority: 2
```

```

    peerIdentifier: peer2
  httpsConfiguration:
    - priority: 1
      peerIdentifier: peer1
    - priority: 2
      peerIdentifier: peer2

```

Note

If required, users can configure more SCP instances in a similar way.

Route-level Configuration

Each route must have configured filters. In case, the SbiRouting functionality is required without the reroutes, then configure `routes[0].metadata.sbiRoutingEnabled=true`, `SbiRouting` in `filterName1`, and set arguments without the `errorHandling` section.

If SbiRouting functionality is required with the reroute mechanism, then configure `routes[0].metadata.sbiRoutingEnabled=true`, `SbiRouting` in `filterName1`, and set arguments with the `errorHandling` section.

The `errorHandling` section contains an array of `errorCriteriaSet` and `actionSet` mapping with priority. The `errorCriteriaSet` and `actionSet` are configured through Helm using `sbiRoutingErrorCriteriaSets` and `sbiRoutingErrorActionSets`.

The `sbiRoutingErrorCriteriaSets` contains an array of `errorCriteriaSet`, where each `errorCriteriaSet` depicts an ID, set of HTTP Methods, set of HTTP Response status codes set of exceptions with `headerMatching` functionality.

The `sbiRoutingErrorActionSets` contains an array of `actionSet`, where each depicts an ID, action to be performed (Currently on REROUTE action is supported) and blacklist configurations.

Following is the SBI routing configuration with the re-route functionality:

Note

Ensure to configure `sbiRoutingErrorCriteriaSets` and `sbiRoutingErrorActionSets`.

The `httpRuriOnly` and `httpsTargetOnly` parameters are used to enable HTTP-Proxy mode communication between Egress Gateway and Peer.

```

- id: nrf_direct
#   uri: https://dummy.dontchange
#   path: /nnrf-disc/**
#   order: 4
#   metadata:

#     httpsTargetOnly: false
#     httpRuriOnly: false
#     sbiRoutingEnabled: false
#   filterName1:
#     name: SbiRouting

```

```
#      args:
#      peerSetIdentifier: set0
#      customPeerSelectorEnabled: false
#      errorHandling:
#      - errorCriteriaSet: scp_direct2_criteria_1
#        actionSet: scp_direct2_action_1
#        priority: 1
#      - errorCriteriaSet: scp_direct2_criteria_0
#        actionSet: scp_direct2_action_0
#        priority: 2
#      - id: scp_route
```

Enable Re-routing

The Re-route mechanism works only for the incoming requests to Egress Gateway that are bound for SBI-Routing. The SBI-Routing bound requests must be re-routed to other instances of SBI based on certain response error codes or exceptions.

Note

The above configuration is effective only when `sbiRoutingEnabled` is set to `true`.

The `errorHandling` section contains an array of `errorCriteriaSet` and `actionSet` mapping with priority. The `errorCriteriaSet` and `actionSet` are configured through Helm using `sbiRoutingErrorCriteriaSets` and `sbiRoutingErrorActionSets`.

Note

`errorCriteriaSet` and `actionSet` must be configured for reroute to work.

To enable re-route functionality with SBIrouting , add the following values in the Helm configuration file:

```
routesConfig:
- id: scp_direct2
  uri: https://dummy.dontchange2
  path: /dummy
  order: 3
  metadata:
    httpsTargetOnly: false
    httpRuriOnly: false
    sbiRoutingEnabled: false
  filterName1:
    name: SbiRouting
    args:
      peerSetIdentifier: set0
      customPeerSelectorEnabled: false
      errorHandling:
        - errorCriteriaSet: scp_direct2_criteria_1
          actionSet: scp_direct2_action_1
          priority: 1
        - errorCriteriaSet: scp_direct2_criteria_0
```

```

        actionSet: scp_direct2_action_0
        priority: 2

sbiRoutingErrorCriteriaSets:
- id: scp_direct2_criteria_0
  method:
    - GET
    - POST
    - PUT
    - DELETE
    - PATCH
  exceptions:
    - java.util.concurrent.TimeoutException
    - java.net.UnknownHostException
- id: scp_direct2_criteria_1
  method:
    - GET
    - POST
    - PUT
    - DELETE
    - PATCH
  response:
    statuses:
      - statusSeries: 4xx
        status:
          - 400
          - 404
      - statusSeries: 5xx
        status:
          - 500
          - 503
  headersMatchingScript: "headerCheck,server,via,.*(SEPP|UDR).*"

sbiRoutingErrorActionSets:
- id: scp_direct2_action_0
  action: reroute
  attempts:2
  blacklist:
    enabled: false
    duration: 60000
- id: scp_direct2_action_1
  action: reroute
  attempts:3
  blacklist:
    enabled: false
    duration: 60000

```

Handling Server and Via Header

This is an enhancement to the SBI routing functionality. An additional alternate routing rule is applied to the Egress Gateway when the header check is included in the configuration. This can be configured through `sbiRoutingErrorCriteriaSet` and corresponding action can be taken by configuring `sbiErrorActionSets`.

To configure SBI Routing with Reroute functionality, see "Enable Rerouting" section.

To enable Server and Via Header handling, add `headersMatchingScript` under the response entity within `sbiRoutingErrorCriteriaSets`.

Note

`headersMatchingScript` is a configuration that accepts a single string with comma separated tokens.

Sample `sbiRoutingErrorCriteriaSets` configuration:

```
sbiRoutingErrorCriteriaSets:
- id: scp_direct2_criteria_1
  method:
    - GET
    - POST
    - PUT
    - DELETE
    - PATCH
  response:
    statuses:
      - statusSeries: 4xx
        status:
          - 400
          - 404
      - statusSeries: 5xx
        status:
          - 500
          - 503
    headersMatchingScript: "headerCheck, server, via, .*(SEPP|UDR).*"

```

The `headersMatchingScript` contains the following tokens:

- `headerCheck` : The Validation function name. It must be constant.
- `server`: Header name
- `Via` : Header Name
- `.*(SEPP|UDR).*` : Regex expression against which the server or via header will be matched against.

This `headersMatchingScript` configuration gets satisfied if the response contains server or via header and the content of the header matches the regex configured. For the criteriaset to be matched, the response method, response status code, and `headersMatchingScript` configuration should be satisfied. The actionset is configured to blacklist the peer if the corresponding criteriaset is matched.

Sample `sbiRoutingErrorActionSets` configuration:

```
sbiRoutingErrorActionSets:
- id: scp_direct2_action_0
  action: reroute
  attempts: 2
  blacklist:

```

```
enabled: true
duration: 60000
```

Once the `sbiRoutingErrorCriteriaSets` is selected, map this actionset to the selected criteriaset in the **errorHandling** section. The corresponding FQDN or Host in the server header value is blacklisted for the duration mentioned in the `blackList` section within the `sbiRoutingErrorActionSets`.

Note

While configuring the `sbiRoutingErrorCriteriaSets` with server header checks (`headersMatchingScript`), ensure that criteriaset has the highest priority in the **errorHandling** section. And, while configuring criteriaset without the server header checks, ensure to keep the `blackList.enabled` as `false`. This is done for server header blacklisting when server header check is required.

3.19 Logging Configuration

This section describes the customizations that you should make in `ocbsf_custom_values_26.1.201.yaml` file to configure logging.

To configure logging in ingress-gateway, configure the following parameters in `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-42 Configurable Parameters for Logging Configuration in Ingress Gateway

Parameter	Description
<code>ingress-gateway.log.level.root</code>	Note: Configure this parameter only when ingress-gateway is enabled. This parameter refers to the Log level for root logs. Default Value: WARN
<code>ingress-gateway.log.level.ingress</code>	Note: Configure this parameter only when ingress-gateway is enabled. This parameter refers to the Log level for ingress logs. Default Value: WARN
<code>ingress-gateway.log.level.oauth</code>	Note: Configure this parameter only when ingress-gateway is enabled. This parameter refers to the Log level for oauth logs. Default Value: WARN

Here is a sample configuration for logging in ingress-gateway in `ocbsf_custom_values_26.1.201.yaml` file:

```
ingress-gateway:
  log:
```

```

level:
  root: WARN
  ingress: WARN
  oauth: WARN

```

Table 3-43 Configurable Parameters for Logging Configuration in Egress Gateway

Parameter	Description
egress-gateway.log.level.root	Note: Configure this parameter only when egress-gateway is enabled. This parameter refers to the Log level for root logs. Default Value: WARN
egress-gateway.log.level.egress	Note: Configure this parameter only when egress-gateway is enabled. This parameter refers to the Log level for ingress logs. Default Value: WARN
egress-gateway.log.level.oauth	Note: Configure this parameter only when egress-gateway is enabled. This parameter refers to the Log level for oauth logs. Default Value: WARN

Here is a sample configuration for logging in egress-gateway in `ocbsf_custom_values_26.1.201.yaml` file:

```

egress-gateway:
  log:
    level:
      root: WARN
      egress: WARN
      oauth: WARN

```

Table 3-44 Configurable Parameters for Logging Configuration in Alternate Route Service

Parameter	Description
alternate-route.log.level.root	This parameter specifies the Log level for root logs. The default value for this parameter is WARN. Note: It is required only when alternate route service is enabled.
alternate-route.log.level.altroute	This parameter specifies the log level for alternate route logs. The default value for this parameter is WARN. Note: It is required only when alternate route service is enabled.

Here is a sample configurations for logging in `ocbsf_custom_values_26.1.201.yaml` file:

```
alternate-route:
```

```
  log:
    level:
      root: WARN
      altroute: WARN
```

Configurations for Debug Tool

At the global level, the `extraContainers` flag can be used to enable or disable injecting extra container, that is, Debug Tool. Users can set `DISABLED` (default value) or `ENABLED` values for this parameter.

The following is a snippet from the `ocbsf_custom_values_26.1.201.yaml` file:

```
# Use 'extraContainers' attribute to control the usage of extra
container(DEBUG tool).
# Allowed Values: DISABLED, ENABLED
extraContainers: DISABLED
```

For more information on Debug Tool, see Oracle Communications Cloud Native Core Binding Support Function Troubleshooting Guide.

To configure label names for Prometheus, you should configure the following configurable parameters in `custom-value.yaml` file:

Table 3-45 Configurable Parameters for Logging Configuration in Prometheus

Parameter	Description	Mandatory/ Optional Parameter	Default Value
<code>tagNamespace</code>	Specifies the tag used for specifying name of the namespace.	Mandatory	<code>kubernetes_n amespace</code> (for CNE 1.8.0) <code>namespace</code> (for CNE 1.9)
<code>tagContainerName</code>	Specifies the tag used for specifying name of the container.	Mandatory	<code>container_na me</code> (for CNE 1.8.0) <code>container</code> (for CNE 1.9)
<code>tagServiceName</code>	Specifies the tag used for specifying name of the service.	Mandatory	<code>kubernetes_n ame</code> (for CNE 1.8.0) <code>service</code> (for CNE 1.9)

The following is a snippet from the `ocbsf_custom_values_26.1.201.yaml` file:

```
#Values for CNE 1.8 {tagNamespace: kubernetes_namespace, tagContainerName:
container_name, tagServiceName: kubernetes_name}
#Values for CNE 1.9 {tagNamespace: namespace, tagContainerName: container,
tagServiceName: service}
tagNamespace: kubernetes_namespace
tagContainerName: container_name
tagServiceName: kubernetes_name
```

3.20 XFCC Header Validation Configuration

This section describes the customizations that you can make in `ocbsf_custom_values_26.1.201.yaml` file to configure XFCC header.

XFCC introduces support for Binding Support Function (BSF) as a producer, to check, if SCP which has sent the HTTP request is the same proxy consumer/client – expected to send an HTTP2 request.

BSF can achieve this by comparing the FQDN of the SCP present in the "x-forwarded-client-cert" (XFCC) of http2 header, with the FQDN of the SCPs configured in the CNC BSF.

To configure XFCC header, you must configure the following parameters in `ocbsf_custom_values_26.1.201.yaml` file:

Table 3-46 Configurable Parameters for XFCC Header Validation Configuration

Parameter	Description
<code>ingress-gateway.xfccHeaderValidation.validation.enabled</code>	This optional parameter determines if incoming xfcc header needs to be validated. Default Value: false
<code>ingress-gateway.xfccHeaderValidation.validation.peerList</code>	Note: Configure this parameter only when xfccHeader validation is enabled. Specifies the list of configured NF FQDN's against which the matchField entry configured, present in the XFCC Header will be validated.
<code>ingress-gateway.xfccHeaderValidation.validation.matchCerts</code>	Note: Configure this parameter only when xfccHeader validation is enabled. This parameter refers to the number of certificates that need to be validated; starting from the right most entry in the XFCC header. - If the parameter is set to -1 (default value), validation is performed against all entries. - If parameter is set to a positive number, validation is performed starting from the right most entry in backwards direction.
<code>ingress-gateway.xfccHeaderValidation.validation.matchField</code>	Note: Configure this parameter only when xfccHeader validation is enabled. This parameter refers to the field in a corresponding XFCC header against which the configured scpList FQDN validation is performed. Default Value: DNS

Table 3-46 (Cont.) Configurable Parameters for XFCC Header Validation Configuration

Parameter	Description
ingress-gateway.xfccHeaderValidation.validation.dnsResolutionInterval	Specifies the interval (in milliseconds) used to resolve failed FQDNs. Default value: 300000
global.xfccHeaderValidation.validation.errorTrigger[i].exceptionType	Specifies the configurable exception or error type for an error scenario in Ingress Gateway. Default value: XFCC_HEADER_INVALID XFCC_MATCHCERTCOUNT_GREATER_THAN_CERTS_IN_HEADER XFCC_HEADER_NOT_PRESENT_OR_EMPTY
global.xfccHeaderValidation.validation.errorTrigger[i].errorCode	Specifies the configurable error code to be returned when the exception or error configured in exceptionType occurs at Ingress Gateway. Default value: 401 402 403
global.xfccHeaderValidation.validation.errorTrigger[i].errorCause	Specifies the configurable error cause to be returned when the exception or error configured in exceptionType occurs at Ingress Gateway. Default value: xfcc header is invalid matchCerts count is greater than the certs in the request xfcc header is not present or empty in the request
global.xfccHeaderValidation.validation.errorTrigger[i].errorTitle	Specifies the configurable error title to be returned when the exception or error configured in exceptionType occurs at Ingress Gateway. Default value: Invalid XFCC Header
global.xfccHeaderValidation.validation.errorTrigger[i].errorDescription	Specifies the configurable error description to be returned when the exception or error configured in exceptionType occurs at Ingress Gateway. Default value: empty string

Here is a sample configurations for XFCC header in ocbsf_custom_values_26.1.201.yaml file:

```
global:
  xfccHeaderValidation:
    validation:
      enabled: false
      peerList:
        - name: scp.com
        - name: smf.com
        - name: amf.com
        - name: scp1.com
      enabled: true
```

```

- name: scp2.com
- name: scp3.com
  enabled: false
- name: xyz.test.com
  enabled: true
  scheme: http
  type: virtual
- name: abc.test.com
  enabled: true
  scheme: https
  type: virtual
- name: xfcc.test.com
  enabled: false
  scheme: http
  type: virtual
matchCerts: -1
matchField: DNS
dnsResolutionInterval: 300000
errorTrigger:
- exceptionType: XFCC_HEADER_INVALID
  errorCode: '401'
  errorCause: xfcc header is invalid
  errorTitle: 'Invalid XFCC Header'
  errorDescription: 'Invalid XFCC Header'
- exceptionType: XFCC_MATCHCERTCOUNT_GREATER_THAN_CERTS_IN_HEADER
  errorCode: '402'
  errorCause: matchCerts count is greater than the certs in the
request
  errorTitle: ''
  errorDescription: ''
- exceptionType: qaZ
  errorCode: '403'
  errorCause: xfcc header is not present or empty in the request
  errorTitle: ''
  errorDescription: ''

```

XFCC Header - Route Level

To enable or disable XFCC header per route, set the `validationEnabled` parameter to `true` under each route (in Ingress Gateway):

```

routesConfig:
- id: reverse_bsf_service
  uri: http://{ template "service-prefix" . }-bsf-management:
{{ .Values.global.servicePorts.bsfManagementServiceHttp }}
  path: /nbsf-management/**
  order: 1
- id: reverse_nrf_notify_service
  uri: http://{ template "service-prefix" . }-nrf-client-nfmanagement:
{{ .Values.global.servicePorts.nrfClientNfManagementHttp }}
  path: /nnrf-client/**
  order: 2

```

Note

These routes are for internal consumption and determine how the incoming traffic is distributed among microservices on the basis of routing properties. To make any modification to these routes other than enabling or disabling XFCC header feature, kindly contact My Oracle Support.

3.21 Aspen service mesh configurations

This section describes the customizations required in `ocbsf_custom_values_26.1.201.yaml` file of Binding Support Function (BSF) to integrate Aspen service mesh with BSF.

- Enable ASM by setting the value for `serviceMeshEnabled` parameter, under global section, as `true`.
- Configure the values for the parameters described in the following table:

Table 3-47 Configurable Parameters for Aspen Servicemesh Configuration

Parameter	Description	Mandatory Parameter	Default Value	Notes
<code>istioSidecarQuitUrl</code>	Specifies the sidecar quit URL (envoy container quite URL) if deployed with serviceMesh. This URL is needed to explicitly shutdown the sidecar container.	Conditional	<code>http://127.0.0.1:15000/quitquitquit</code>	Applicable only when <code>serviceMeshCheck</code> parameter is set to <code>true</code> .
<code>istioSidecarReadyUrl</code>	Specifies the sidecar ready URL (envoy container quite URL) if deployed with serviceMesh. This URL is needed to check the readiness of the sidecar container during initialization process. The gateway container will come up only after sidecar container is ready.	Conditional	<code>http://127.0.0.1:15000/ready</code>	Applicable only when <code>serviceMeshCheck</code> parameter is set to <code>true</code> .

- In the global section, uncomment the following annotations to include port 9000 - a Prometheus scrap port

```
allResources:
  labels: {}
  annotations: {
    #Enable this section for service-mesh based installation
    # traffic.sidecar.istio.io/excludeInboundPorts: "9000",
    # traffic.sidecar.istio.io/excludeOutboundPorts: "9000"
  }
```

- (Optional) If BSF is deployed with OSO, the pods need to have an annotation **oracle.com/cnc: true**.

```

lbServices:
  labels: {}
  annotations: {}

lbDeployments:
  labels: {}
  annotations: {}
  # The annotation oracle.com/cnc: "true" is required if OSO is
used
  #oracle.com/cnc: "true"
  #sidecar.istio.io/inject: "true"
  #sidecar.istio.io/rewriteAppHTTPProbers: "true"

nonlbServices:
  labels: {}
  annotations: {}

nonlbDeployments:
  labels: {}
  annotations: {}
  # The annotation oracle.com/cnc: "true" is required if OSO is
used
  #oracle.com/cnc: "true"
  #sidecar.istio.io/inject: "true"
  #sidecar.istio.io/rewriteAppHTTPProbers: "true"

```

- Uncomment the following annotations in the deployment sections of nrf-client-nfdiscovery, nrf-client-nfmanagement, diam-gateway, ingress-gateway, egress-gateway, and alternate-route services

```

deployment:
  customExtension:
    annotations: {
      #Enable this section for service-mesh based
installation:
  #      traffic.sidecar.istio.io/excludeOutboundPorts:
"9000,8095,8096,7,53",
  #      traffic.sidecar.istio.io/excludeInboundPorts:
"9000,8095,8096,7,53"
  }

```

Here, 8095 and 8096 are Coherence ports.

① Note

Port 53 is included only if DNS lookup bypasses the sidecar connection management.

- **Disable init containers:** Init containers do not work when the namespace has aspen service mTLS enabled. To disable init containers, set the value for `initContainerEnable` to false in custom values file.

```
global:
  initContainerEnable: false
```

- **PERMISSIVE rule:** To set Permissive rule for Diameter Gateway and Ingress Gateway Service, set the following flags to true in `ocbsf_custom_values_26.1.201.yaml` file:

```
global:
  istioIngressTlsSupport:
    diamGateway: false
```

- **Configuring Sidecar Resources for Microservices:**
Resource limits for Istio sidecar containers can be configured either globally (for all services) or on a per-service basis. This is managed in the `ocbsf_custom_values_26.1.201.yaml` file.

Set the following Aspen Service Mesh (ASM) annotations to define resource allocation for sidecar containers:

- **sidecar.istio.io/proxyCPU:** CPU requested by the sidecar.
- **sidecar.istio.io/proxyCPULimit:** CPU limit for the sidecar.
- **sidecar.istio.io/proxyMemory:** Memory requested by the sidecar.
- **sidecar.istio.io/proxyMemoryLimit:** Maximum memory limit for the sidecar.

Configuration Methods:

- **Global Configuration:**
Specify these annotations at the global level to apply the same resource settings to all microservices in the deployment.

Example:

```
global:
  customExtension:
    allResources:
      labels: {}
      annotations: {
        sidecar.istio.io/proxyCPU: '2',
        sidecar.istio.io/proxyCPULimit: '2',
        sidecar.istio.io/proxyMemory: '128Mi',
        sidecar.istio.io/proxyMemoryLimit: '1Gi'
      }
```

- **Service-level Configuration:**
Alternatively, specify the annotations within each individual service's configuration to provide customized resource limits per service.

Example

```
<service>:
  deployment:
    customExtension:
      annotations: {
        sidecar.istio.io/proxyCPU: '4',
        sidecar.istio.io/proxyCPULimit: '4',
        sidecar.istio.io/proxyMemory: '128Mi',
```

```
sidecar.istio.io/proxyMemoryLimit: '1Gi'
}
```

- Apply these settings to ensure optimal resource allocation for sidecar containers according to you deployment's requirements.

3.22 Alternate Route Service Configuration

This section describes how to configure alternate route service (DNS-SRV) by customizing parameters in the `ocbsf_custom_values_26.1.201.yaml` file.

Note

Users must customize parameters, described in this section, only when alternate route service is enabled.

With SRV Records, you can configure and maintain NF FQDN dynamically at the DNS Server, which can be further selected by Cloud Native Core Binding Support function, when there is a network function failure. It is achieved by performing a SRV query on the virtual FQDN configured at the BSF, instead of configuring primary and secondary NRF statically in every CNC BSF, only during instantiation time. This option of DNS lookup for SRV records would also provide alternate NFs to the BSF during failover.

To configure DNS-SRV, you should configure the following configurable parameters in `custom-value.yaml` file:

Table 3-48 Configurable Parameters for Alternate Route Service Configuration

Parameter	Description
<code>global.alternateRouteServiceEnable</code>	This global parameter describes whether to enable or disable Alternate Route service during Helm deployment. By default, the value for this parameter is set to true.
<code>alternate-route.staticVirtualFqdns[0].name</code>	This optional parameter describes the name of the virtual FQDN/FQDN.
<code>alternate-route.staticVirtualFqdns[0].alternateFqdns[0].target</code>	This parameter describes the name of the alternate FQDN mapped to the virtual FQDN - described in the previous row. Note: Users must define the value of this parameter if <code>staticVirtualFqdns[0].name</code> is defined.
<code>alternate-route.staticVirtualFqdns[0].alternateFqdns[0].port</code>	This parameter describes the port number of the alternate FQDN. Note: Users must define the value of this parameter if <code>staticVirtualFqdns[0].name</code> is defined.
<code>alternate-route.staticVirtualFqdns[0].alternateFqdns[0].priority</code>	This parameter describes the priority of the alternate FQDN. Note: Users must define the value of this parameter if <code>staticVirtualFqdns[0].name</code> is defined.
<code>alternate-route.dnsSrvEnabled</code>	This parameter describes whether to enable or disable the DNS-SRV query to coreDNS Server. By default, the value is set to true.

Table 3-48 (Cont.) Configurable Parameters for Alternate Route Service Configuration

Parameter	Description
alternate-route.dnsSrvFqdnSetting.enabled	This parameter describes whether to enable or disable the usage of custom pattern for the FQDN while triggering DNS-SRV query. By default, the value is set to true. Note: If this flag is set to false, then default value: "{scheme}._tcp.{fqdn}." will be used.
alternate-route.dnsSrvFqdnSetting.pattern	This parameter describes the pattern of the FQDN that is used to format the incoming FQDN and Scheme while triggering DNS-SRV query. The default value for this parameter is "{scheme}._tcp.{fqdn}." Note: Users must define the value of this parameter if dnsSrvFqdnSetting.enabled is set to true.
egress-gateway.dnsSrv.port	This parameter describes the port of DNS Alternate Route Service. Default Value: *svcAlternateRouteServiceHttp Note: Users must define the value of this parameter if DnsSrv is required.
nrf-client-nfmanagement.alternateRouteServiceEnabled	This parameter notifies nrf-client services if alternate route service is deployed or not. By default, this parameter is set to false. Note: Users must set this parameter to true if global.alternateRouteServiceEnable is set to true.
nrf-client-nfdiscovery.alternateRouteServiceEnabled	This parameter notifies nrf-client services if alternate route service is deployed or not. By default, this parameter is set to false. Note: Users must set this parameter to true if global.alternateRouteServiceEnable is set to true.
alternate-route.minReplicas	Specifies the minimum replicas to scale to maintain an average CPU utilization.
alternate-route.maxReplicas	Specifies the maximum replicas to scale to maintain an average CPU utilization.

Here is a sample configurations for DNS-SRV in ocbsf_custom_values_26.1.201.yaml file:

```
#Static virtual FQDN Config
staticVirtualFqdns:
  - name: https://abc.test.com
    alternateFqdns:
      - target: abc.test.com
        port: 5060
        priority: 10
      - target: xyz.test.com
        port: 5060
        priority: 20
  - name: http://xyz.test.com
    alternateFqdns:
      - target: xyz.test.com
        port: 5060
        priority: 10
      - target: abc.test.com
        port: 5060
        priority: 20 #Flag to control if DNS-SRV queries are sent to
coreDNS or not
```

```

    dnsSrvEnabled: true
    #Below configuration is for customizing the format of FQDN which will used
    while querying coreDNS for SRV Records
    dnsSrvFqdnSetting:
        enabled: true #If this flag is disabled, then default value of
        "{scheme}._tcp.{fqdn}." will be used for Pattern
        pattern: "{scheme}._tcp.{fqdn}." #Ex: _http._tcp.service.example.org.

egress-gateway:
  dnsSrv:
    host: 10.75.225.67
    port: 32081

```

3.23 Additional Configurations

- **Annotation to support custom extension global parameters:** To support custom extension global parameters, update the following parameters in custom extension under global section of `ocbsf_custom_values_26.1.201.yaml` file:

```

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

    lbServices:
      labels: {}
      annotations: {}

    lbDeployments:
      labels: {}
      annotations: {}

    nonlbServices:
      labels: {}
      annotations: {}

    nonlbDeployments:
      labels: {}
      annotations: {}

```

- **Annotation to support OSO:** To deploy BSF with OSO, you must add the following annotation to the custom extension under global section of `ocbsf_custom_values_26.1.201.yaml` file:

```

global:
  customExtension:
    lbDeployments:
      annotations:
        oracle.com/cnc: "true"

    nonlbDeployments:

```

```
annotations:  
  oracle.com/cnc: "true"
```

Note

After helm install is complete, all the nodes should have the above mentioned notation.

- **Custom container name:** You can customize the name of containers of a pod with a prefix and suffix. To do so, add the prefix and suffix to the k8sResource under global section of ocbsf_custom_values_26.1.201.yaml file:

```
global:  
  k8sResource:  
    container:  
      prefix: ABC  
      suffix: XYZ
```

Then, after installing BSF, you will see the container names as shown below:

```
Containers:  
  abcd-am-service-xyz:
```

- **Kubernetes service account name:** You can use a custom service account for all services by adding it to appinfo section in the ocbsf_custom_values_26.1.201.yaml file:

```
appinfo:  
  serviceAccountName: ocbsfsaccount
```

Note

You can create the service account and roles before the installation as well.

AppInfo Configurations

The following table describes the configurable parameters for AppInfo service:

Table 3-49 Customizable Parameters for Common Configuration Service in appinfo

Parameter	Description	Default Value	Notes
appinfo.watchMySQL	If the value for this parameter is set to true, appinfo periodically queries local DB status from the db monitor service specified by dbStatusUri. The DB monitor service returns 200 if the database is healthy, and 503 if database is not usable. If DB status is not good, then appinfo will inform nrfclient to mark PCF suspended.	false	When its value is set to true, the user must specify dbStatusUri.
appinfo.replicationStatusCheck	When the value for this parameter is set to true, then appinfo periodically queries the replication status from the db monitor service specified by replicationUri. This value is then used by NRF.	false	When its value is set to true, the user must specify replicationUri.
appinfo.dbStatusUri	Specifies the URI provided by the DB monitor service to query local database status. Example: http://occne-db-monitor-svc.occne-infra:8080/db-tier/status/local	empty string	
appinfo.realtimeDbStatusUri	Specifies the URI provided by the realtime DB monitor service to query the status of the realtime DB pointing to the cluster. http://occne-db-monitor-svc.occne-infra:8080/db-tier/status/cluster/local/realtime	empty string	
appinfo.replicationUri	Specifies the URI provided by the DB monitor service to query replication status. Example: http://occne-db-monitor-svc.occne-infra:8080/db-tier/status/replication	empty string	
appinfo.commonCfgClient.enabled	Specifies whether to enable or disable dynamic logging using common configuration service.	true	

Table 3-49 (Cont.) Customizable Parameters for Common Configuration Service in appinfo

Parameter	Description	Default Value	Notes
appinfo.commonCfgServer.port	Specifies the port of common configuration server.	8000	Same value as servicePorts.cmS serviceHttp.
appinfo.dbConfig.dbHost	Specifies the Hostname of MySQL that is used to store configurations.	Not applicable	Same value as global.envMySQLHost.
appinfo.dbConfig.dbPort	Specifies the port number of MySQL.	Not applicable	Same value as global.envMySQLPort.
appinfo.dbConfig.secretName	Specifies the database secret from which the db name, db password and db user name is picked.	occnp-db-pass	Same value as global.dbCredSecretName.
appinfo.dbConfig.dbName	Specifies the database name to be used to store the common configuration.	occnp_commonconfig	
appinfo.dbConfig.dbUNameLiteral	Specifies the database literal name that shall be used as per the <dbConfig.secretName>.	mysql-username	
appinfo.dbConfig.dbPwdLiteral	Specifies the database password literal name that shall be used as per the <dbConfig.secretName>.	mysql-password	
appinfo.dbTierVersionUri	Specified the URI provided by the replication service to query Db tier version. For example:http://mysql-cluster-sitea-siteb-replication-svc/db-tier/version	empty string	Before Enabling Infra Validate flag Customers are suggested to make sure that there are no critical alarms before upgrading/installing a new release in order to avoid failures. Also, make sure that replication is up.

3.24 Configurations for Metrics

Global Metrics Configurations

Starting with CNE 1.9.0, if the user wants to enable monitoring via Prometheus, the following parameters must be configured:

Table 3-50 Global Configurations for Metrics

Parameter	Description
cncMetricsName	This parameter specifies the port, that is, <code>cnc-metrics</code> that Prometheus will scrape on.
exposeObservabilityAtService	This parameter specifies whether to enable or disable Prometheus monitoring of services. By default, the value is set to false and services are not captured in Prometheus GUI.

You can add prefix and suffix to metrics for BSF services by using the following parameters:

```
metricPrefix: &metricPrefix 'ocbsf'
metricSuffix: &metricSuffix ''
```

Table 3-51 Prefix and Suffix for Metrics

Parameter	Description
metricPrefix	This parameter specifies the prefix that you want to add to the metrics for BSF services. Default value: <code>ocbsf</code>
metricSuffix	This parameter specifies the suffix that you want to add to the metrics for BSF services. Default value: empty string

A reference is made to the `metricPrefix` and `metricSuffix` parameters, defined in the global section, under `nrf-client-nfdiscovery` and `nrf-client-nfmanagement` configurations.

Note

- If you choose to customize prefix, then it is required to align the NF delivered Grafana charts and Prometheus alerts with the updated metric names.
- When you define a suffix for metrics, it may happen that the suffix appears in the middle of the metric name, and not towards the end. This is due to the fact that Micrometer library autogenerates some metrics and adds a suffix after the user-defined suffix.
Example: If you define suffix as `ocbsf`, then the resulting metric name would appear in the system as `http_in_conn_response_ocbsf_total`.

3.25 Overload Manager Configurations

This section describes the customizations that can be done in `ocbsf_custom_values_26.1.201.yaml` file to configure Overload Manager feature under `perf-info`.

Table 3-52 Configurable Parameters for overload Manager Configuration in Perf-Info

Parameter	Description	Mandatory/ Optional Parameter	Default Value
perf-info.overloadManager.enabled	Specifies whether to enable or disable overload reporting.	Optional	false
perf-info.envMySQLDatabase	Specifies the name of the database used for overload management. For georedundant setup, the value for this parameter must be unique for each site.	Conditional Note: This parameter value is required if the overload manager functionality is enabled by setting the value of perf-info.overloadManager.enabled to true.	
perf-info.overloadManager.ingressGatewaySvcName	Specifies the names of backend services		ocbsf-ingress-gateway
perf-info.overloadManager.ingressGatewayPort	Specifies the port number of Ingress Gateway	Mandatory	80
perf-info.overloadManager.nfType	Specifies the NF type that is used to query configuration from common configuration server.		BSF

Here is a sample overloadManager configurations in perf-info in ocbsf_custom_values_26.1.201.yaml file:

```
perf-info:
  configmapPerformance:
    prometheus: ''
    # envMySQLDatabase is used for overload management.
    # If the customer does not use the overload management feature, this can be
    ignored.
    envMySQLDatabase: ''
  overloadManager:
    enabled: false
    # nfType is used to query configuration from common cfg server
    nfType: BSF
```

3.26 Configurable Error Codes

This section describes the parameters that you can customize for configurable error codes.

Table 3-53 Configurable Parameters for Error Codes - Global

Parameter	Description	Mandatory/ Optional Parameter	Default Value
configurableErrorCodes.enabled	Specifies whether to enable or disable configurable error codes that can be used for messages over Ingress Gateway and Egress Gateway.	Optional	false

Table 3-54 Configurable Parameters for Error Codes - Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
ingress-gateway.configurableErrorCodes.enabled	Specifies whether to enable or disable configurable error codes that can be used for messages over Ingress Gateway and Egress Gateway.	Optional	false

For a given error scenario, you can define exceptionType, errorCode, errorDescription, errorCause, and errorTitle as shown in the following snippet from the `ocnp_custom_values_CNC Policy.yaml` file.

Following is the configuration for error codes at global level:

```
ingress-gateway:
  configurableErrorCodes:
    enabled: true
    errorScenarios:
      - exceptionType: "XFCC_HEADER_INVALID"
        errorProfileName: "ERR_1300"
      - exceptionType: "XFCC_HEADER_VALIDATION_FAILURE"
        errorProfileName: "ERR_1300"

  errorCodeProfiles:
    - name: ERR_1300
      errorCode: 401
      errorCause: "xfcc header is invalid"
      errorTitle: "Invalid XFCC Header"
      errorDescription: "Invalid XFCC Header"
```

Following points must be noted for the global level configuration:

- To enable configurable error code global configurableErrorCodes flag must be set to true. If this flag is false then the hardcoded error codes will be returned when an exception is encountered at Ingress and Egress Gateways.
- If global configurableErrorCodes flag is set to true then atleast one entry must be configured in the errorScenarios section.
- For every Exception in errorScenarios there must be an error profile with that exceptionType. Moreover, a profile with that name must be configured in errorCodeProfiles section example - if errorProfileName: "ERR_1300" has been configured then a profile with name ERR_1300 must be present in errorCodeProfiles section.

- ExceptionType field in global and in the routes section is non configurable. These are hard coded values and can be taken from custom.yaml file.

Following is the configuration for error codes at route level:

```
routesConfig:
- id: route1
  uri:
  path: /dummy/*/dummies
  order: 1
  method: POST
  metadata:
    configurableErrorCodes:
      enabled: true
      errorScenarios:
        - exceptionType: "XFCC_HEADER_INVALID"
          errorProfileName: "ERR_1300"
        - exceptionType: "XFCC_HEADER_VALIDATION_FAILURE"
          errorProfileName: "ERR_1300"
```

Following points must be noted for the route level configuration:

- If Route level is enabled, it has higher precedence over global level.
- For Route level configurable error codes to work, configurableErrorCodes flag must be set to true both at route level as well as global level.
- For a given exception at gateway, if there is no match at route level then global level is matched. If there is no match at global level, then hardcoded error values are returned.
- If configurableErrorCodes flag is disabled for a specific route and if an exception occurs at that route then hardcoded error responses will be returned irrespective of what is defined at global level.

Note

For every errorScenario, exceptionType and errorCode are mandatory parameter configurations.

Configurable Error Codes - SCP Integration

The following parameters are added under Egress Gateway for SCP related configurations. These error code configurations are included in error response from Egress Gateway when it is unable to resolve DNS successfully:

```
dnsSrv:
  port: *svcAlternateRouteServiceHttp
```

For more information about the error codes, see [Configurable Error Codes](#).

3.27 Server Header Configurations

This section describes the parameters that you can configure to enable support for server header at Ingress Gateway.

Table 3-55 Configurable Parameters for Server Header at Ingress Gateway

Parameter	Description	Default Value
ingress-gateway.serverHeaderConfigMode	This optional parameter specifies the mode of operation for configuring server header configuration. The possible values this parameters can be set to is: • REST • HELM Based on the value assigned to this parameter, the feature flag for "serverheaderdetails" must be enabled either in Rest or Helm configurations respectively For more information, see "Server Header Support on Ingress Gateway" section in <i>Oracle Communications Cloud Native Core, Binding Support Function REST Specification Guide</i>.	REST

The following is a snippet from the `ocbsf-24.1.0-custom-values.yaml` file:

```
ingress-gateway:
  serverHeaderConfigMode: REST # Possible values: HELM, REST. Based on this
  value, the feature flag for "server" header will need to be enabled either
  in Helm configuration or Rest configuration.
```

3.28 Creating Custom Headers

This section provides information on how to create custom headers for routes in BSF.

You can customize the headers present in the requests and responses based on the type of HTTP methods. This framework modifies the outgoing request or response by adding a new header either with a static value or with a value based on incoming request or response headers at entry or exit points.

By setting the `override` attribute value as `true`, you can override the existing headers. It is an optional attribute. It adds a new header or replaces the value of an existing header if one of the value is mapped to the source header. The value of this attribute is `false` by default.

The following is a sample configuration for custom header in `bsf_management_deregister`:

```
- id: bsf_management_deregister
  uri: http://{ template "service-name-bsf-management" . } }:
{{ .Values.global.servicePorts.bsfManagementServiceHttp }}
  path: /nbsf-management/**
  order: 2
  method: DELETE
  filters:
    customReqHeaderEntryFilter:
      headers:
        - methods:
            - DELETE
          headersList:
            - headerName: 3gpp-Sbi-Message-Priority
              defaultVal: 18
              source: incomingReq
```

```
sourceHeader: 3gpp-Sbi-Message-Priority  
override: false
```

Note

The attributes `headerName` and `sourceHeader` are case sensitive. Ensure that the value is same as in the incoming request or response in order to extract values from or override value of any particular header.

3.29 Ingress Gateway Readiness Probe Configuration

This section describes the readiness probe configurations in the Ingress Gateway.

Ingress Gateway uses the readiness logic provided by Kubernetes to determine if a pod can accept or reject the incoming requests.

This feature enhances the readiness logic to determine the status of the pod. You can configure the feature in BSF only through Helm. Based on the configurations, further checks are performed to determine the health of the pod.

An in-memory cache is maintained to store the updated configuration. The cache is updated if a profile is modified, added, or deleted. Ingress gateway periodically makes a GET request to the URLs that are configured using a scheduler that runs in the background. If the GET request is successful, then other checks can take place. Otherwise, the pod is marked as unhealthy.

Note

If there are any pending requests waiting for the response and readiness state of pod changes from `READY` to `NOT_READY`, then these requests are not considered.

The following table describes the parameters for configuring Readiness Probe in Ingress Gateway:

Table 3-56 Configurable Parameters for Readiness Probe Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
<code>readinessConfigMode</code>	Specifies the mode to configure Readiness Probe in Ingress Gateway.	Mandatory	HELM	
<code>readinessCheckEnabled</code>	Specifies whether to enable or disable Readiness Probe in Ingress Gateway.	Mandatory	false	

Table 3-56 (Cont.) Configurable Parameters for Readiness Probe Configuration

Parameter	Description	Mandatory/Optional Parameter	Default Value	Notes
readinessIndicatorPollingInterval	Specifies the time (in milliseconds) at which the Readiness Cache updates the readiness status of Ingress Gateway performing the probe or setting the readiness state value to <code>onExceptionUsePreviousState</code> .	Mandatory	3000	
readinessConfig.serviceProfiles.id	Specifies the ID of the profile.	Mandatory	Readiness-profile-DBStatus	
readinessConfig.serviceProfiles.url	Specifies the URL to which the Readiness Probe is sent out to retrieve a response, on the basis of which the state of the Ingress Gateway pod will be decided.	Mandatory	http://{{ template "service-name-app-info" . }}:{{ .Values.global.containerPorts.appInfoHttp }}/status/category/realtimedatabase	In addition to the default value, you can use the following values: 1. FQDN/IP Address. 2. Any microservice to define dependency upon: <code>http://<HelmReleaseName>-<BSFServiceName>:9000/actuator/health/readiness</code>
readinessConfig.serviceProfiles.responseCode	Specifies the response code expected from the service. If the actual response code matches with the configured one then pod will be marked as healthy.	Mandatory	200	

Table 3-56 (Cont.) Configurable Parameters for Readiness Probe Configuration

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Notes
readinessConfig.serviceProfiles.responseBody	Specifies the response expected from the service. If the actual response matches with the configured one then pod will be marked as healthy.	Mandatory	Running	
readinessConfig.serviceProfiles.onExceptionUsePreviousState	Specifies whether to use the previous state of Ingress Gateway. When this flag is set to true, response and responseCode checks are not made irrespective of the previous state of service on Ingress Gateway.	Mandatory	true	
readinessConfig.serviceProfiles.initialState	Specifies the initial state to be specified. It can be either ACCEPTING_TRAFFIC (to accept all incoming requests) or REFUSING_TRAFFIC (to reject all incoming requests).	Mandatory	ACCEPTING_TRAFFIC	
readinessConfig.serviceProfiles.requestTimeout	Specifies the timeout value of the probe in milliseconds.	Optional	2000	

Check the following when the Ingress Gateway pod comes up:

1. If the service profiles are not configured, then the readiness probe of Ingress Gateway fails and the pod is marked as unhealthy.
2. If the service profiles are configured, check the mandatory parameters: `id`, `url`, `onExceptionUsePreviousState`, and `initialState` for their validity. If they are invalid, then the pod is marked as unhealthy.

Note

You must configure one of these parameters: `responseBody` or `responseCode` in the service profile. If any of these checks fail, then the pod does not come up in the case of Helm based configuration.

3. If there is any error like connection failure or connection timeout during making a request to backend service, then `onExceptionUsePreviousState` attribute is checked. If it is set to true, then previous state is used for that URL. If previous state is unavailable, then initial state is used. If `onExceptionUsePreviousState` is false, then the pod is marked as unhealthy.

3.30 Late Arrival Handling Configurations

This section describes the parameters that user can configure for late arrival handling feature.

Table 3-57 Configurable Parameters for Late Arrival Handling at Ingress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Applicable to Deployment	Added/ Deprecated/ Updated in Release
ingress-gateway.isSbiTimerEnabled	Specifies whether to enable or disable SBI timer header enhancement. If the value of this parameter is set to true, SBI headers (3gpp-Sbi-Sender-Timestamp, 3gpp-Sbi-Max-Rsp-Time, and 3gpp-Sbi-Origination-Timestamp) are used along with route level (if configured) and global level request timeout to calculate final request timeout. After calculating the final request timeout, original values of 3gpp-Sbi-Sender-Timestamp, 3gpp-Sbi-Max-Rsp-Time and 3gpp-Sbi-Origination-Timestamp are published in custom headers Orig-3gpp-Sbi-Sender-Timestamp, Orig-3gpp-Sbi-Max-Rsp-Time and Orig-3gpp-Sbi-Origination-Timestamp respectively. If the value for this parameter is set to false, SBI headers are not taken into consideration even if they are present and no custom headers are published.	Optional	false	CNC BSF, CNC Policy & PCF	Added in Release 1.15.0
ingress-gateway.publishHeaders	Specifies if the originating headers shall be populated and sent to the backend.	Optional	false	CNC BSF, CNC Policy & PCF	Added in Release 1.15.0
ingress-gateway.sbiTimerTimezone	Specifies the time zone. It can be either set to GMT or ANY. If it is set to GMT then, the GMT should be specified in the header. If it is not specified, the time zone is assumed as GMT. If it is set to ANY then, the required time zone must be specified in the header. The timeout calculation is made as per the time zone specified in the header. If time zone is not specified then, the request is rejected and a gauge metric is pegged.	Optional	GMT	CNC BSF, CNC Policy, PCF, & PCRF	Added in Release 1.15.0

The following is a snippet from the `occnp-1.15.0-custom-values.yaml` file:

```
# Late arrival handling
isSbiTimerEnabled: false
publishHeaders: false
sbiTimerTimezone: GMT
```

Table 3-58 Configurable Parameters for Late Arrival Handling at Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value	Applicable to Deployment	Added/ Deprecated/ Updated in Release
egress-gateway.isSbiTimerEnabled	Specifies whether to enable or disable SBI timer header enhancement. If the value of this parameter is set to true, SBI headers (3gpp-Sbi-Sender-Timestamp, 3gpp-Sbi-Max-Rsp-Time, and 3gpp-Sbi-Origination-Timestamp) are used along with route level (if configured) and global level request timeout to calculate final request timeout. After calculating the final request timeout, original values of 3gpp-Sbi-Sender-Timestamp, 3gpp-Sbi-Max-Rsp-Time and 3gpp-Sbi-Origination-Timestamp are published in custom headers Orig-3gpp-Sbi-Sender-Timestamp, Orig-3gpp-Sbi-Max-Rsp-Time and Orig-3gpp-Sbi-Origination-Timestamp respectively. If the value for this parameter is set to false, SBI headers are not taken into consideration even if they are present and no custom headers are published.	Optional	false	CNC BSF, CNC Policy & PCF	Added in Release 1.15.0
egress-gateway.sbiTimerTimezone	Specifies the time zone. It can be either set to GMT or ANY. If it is set to GMT then, the GMT should be specified in the header. If it is not specified, the time zone is assumed as GMT. If it is set to ANY then, the required time zone must be specified in the header. The timeout calculation is made as per the time zone specified in the header. If time zone is not specified then, the request is rejected and a gauge metric is pegged.	Optional	GMT	CNC BSF, CNC Policy & PCF	Added in Release 1.15.0
egress-gateway-ignoreMaxRspTimeHeader	Specifies whether to ignore <i>3gpp-Max-Rsp-Time</i> while calculating the final request timeout.	Optional	false	CNC BSF, CNC Policy & PCF	Added in Release 1.15.0

To create the collision headers that are used for collision detection in BSF it is necessary to add the following configuration to BSF Ingress Gateway.

```

routesConfig:
  - id: bsf_management_register
    uri: http://{ template "service-name-bsf-management" . } }:
  {{ .Values.global.servicePorts.bsfManagementServiceHttp }}
    path: /nbsf-management/**
    order: 1
    method: POST
    filters:
      customReqHeaderEntryFilter:
        headers:
          - methods:

```

```

- POST
headersList:
  - headerName: 3gpp-Sbi-Message-Priority
    defaultVal: 24
    source: incomingReq
    sourceHeader: 3gpp-Sbi-Message-Priority
    override: false
  - headerName: collision-3gpp-origination-timestamp
    source: incomingReq
    sourceHeader: 3gpp-Sbi-Origination-Timestamp
    override: false
  - headerName: collision-custom-sender-timestamp
    source: incomingReq
    sourceHeader: Custom-Sbi-Sender-Timestamp
    override: false

- headerName: collision-3gpp-sender-timestamp
  source: incomingReq
  sourceHeader: 3gpp-Sbi-Sender-Timestamp
  override: false

```

3.31 Controlled Shutdown Configurations

This section describes the customizations that can be done in `ocbsf_custom_values_26.1.201.yaml` file to configure controlled shutdown feature.

Table 3-59 Global Parameter for Controlled Shutdown

Parameter	Description	Mandatory/ Optional Parameter	Default Value
<code>global.enableControlledShutdown</code>	Specifies whether to enable or disable the Controlled Shutdown feature.	Mandatory	False

Table 3-60 Configurable Parameters for Controlled Shutdown in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
<code>egress-gateway.errorcodeprofiles</code>	Error defined by the user	Optional	NA
<code>egress-gateway.errorcodeprofiles.name</code>	Name of the error profile	Optional	NA
<code>egress-gateway.errorcodeprofiles.errorCode</code>	Error code of the error profile	Optional	NA
<code>egress-gateway.errorcodeprofiles.errorCause</code>	Cause of the error profile	Optional	NA
<code>egress-gateway.errorcodeprofiles.errorTitle</code>	Title of the error profile	Optional	NA

Table 3-60 (Cont.) Configurable Parameters for Controlled Shutdown in Egress Gateway

Parameter	Description	Mandatory/ Optional Parameter	Default Value
egress-gateway.errorcodeprofiles.errorDescription	Description of the error profile	Optional	NA
egress-gateway.routesConfig	Routes configuration processed by the Egress Gateway	Optional	NA
egress-gateway.routesConfig.id	ID of the route	Optional	NA
egress-gateway.routesConfig.uri	URI of the route	Optional	NA
egress-gateway.routesConfig.path	Path of the route	Optional	NA
egress-gateway.routesConfig.order	Order in which the routes will be processed	Optional	NA
egress-gateway.routesConfig.filters	Conditions on the routes	Optional	NA
egress-gateway.routesConfig.filters.controlledShutdownFilter	Filter specified for Controlled Shutdown feature	Optional	NA
egress-gateway.routesConfig.filters.controlledShutdownFilter.applicableShutdownStates	States of Controlled shutdown feature, that is COMPLETE_SHUTDOWN	Optional	NA
egress-gateway.routesConfig.filters.controlledShutdownFilter.unsupportedOperations	Operations which needs not be supported for controlled shutdown feature	Optional	NA
egress-gateway.controlledShutdownErrorMapping	Array containing route ID and error profile name	Optional	NA
egress-gateway.controlledShutdownErrorMapping.routeErrorProfileList	List of route ID and their corresponding error profile names	Optional	NA
egress-gateway.controlledShutdownErrorMapping.routeErrorProfileList.routeId	Route ID on which the error profile name needs to be mapped	Optional	NA
egress-gateway.controlledShutdownErrorMapping.routeErrorProfileList.errorProfileName	Error name from the error code profiles to be mapped in route ID	Optional	NA

Here is a sample Error Codes configuration in Egress Gateway in the `ocbsf_custom_values_26.1.201.yaml` file:

```
errorcodeprofiles:
  - name: error300,
    errorCode: 300,
    errorCause: "",
    errorTitle: "",
    retry-after: "",
```

```

    errorDescription: ""
  - name: error500,
    errorCode: 500,
    errorCause: "",
    errorTitle: "",
    retryAfter: "",
    errorDescription: ""

```

Here is a sample routes configuration for Controlled Shutdown in Egress Gateway in the `ocbsf_custom_values_26.1.201.yaml` file:

```

routesConfig:
  - id: nrf_state
    uri: https://dummy.dontchange_1
    path: /nnrf-nfm/*
    order: 1
  - id: sampleRoute
    uri: https://dummy.dontchange_2
    path: /**
    order: 2
    metadata:
      httpsTargetOnly: false
      httpRuriOnly: false
      sbiRoutingEnabled: true
      oauthEnabled: false
    filterNameControlShutdown:
      name: ControlledShutdownFilter
      args:
        applicableShutdownStates:
          - COMPLETE_SHUTDOWN
        unsupportedOperations:
          - GET
          - PUT
          - PATCH
          - POST
          - DELETE

```

Here is a sample Error Codes Mapping configuration in Egress Gateway in the `ocbsf_custom_values_26.1.201.yaml` file:

```

controlledShutdownErrorMapping:
  routeErrorProfileList:
    - routeId: sampleRoute
      errorProfileName: "error503"

```

3.32 Common Configurations for Services

This section describes the configurable parameters that can be used to perform some common configurations applicable to different services while deploying BSF.

Common Reference Configurations

You can configure some common parameters that are used in multiple services by configuring `commonRef` section under `global` parameters section of the Custom Values YAML file. The

parameter values can be set under `commonRef` and same value is used by all the services through the reference variable for the configuration.

The following section describes the `commonRef` parameters for common configuration:

Table 3-61 Common Reference Configurations

Parameter	Description	Mandatory Parameter	Default Value	Notes
<code>&configServerImage</code>	Specifies the name of the config server container image.	Yes	oc-config-server	
<code>&configServerDB</code>	Specifies the name of the config server database.	Yes	ocbsf_config_server	
<code>&commonConfigDB</code>	Specifies the name of the common config database.	Yes	ocbsf_commonconfig	
<code>commonCfgSvc.commonCfgServer.port</code>	Specifies the common config server port for common config service.	Yes	8000	Same value as <code>global.servicePorts.cmsServiceHttp</code> .
<code>&dbCommonConfig.dbHost</code>	Specifies the MySQL database host for services.	Yes		Same value as <code>global.envMySQLHost</code> .
<code>&dbCommonConfig.dbPort</code>	Specifies MySQL database port for services.	Yes		Same value as <code>global.envMySQLPort</code> .
<code>&dbCommonConfig.dbName</code>	Specifies common config database name for services to store common configurations.	Yes	ocbsf_commonconfig	Same value as <code>global.commonRef.commonConfigDB</code>
<code>&dbCommonConfig.dbUserNameLiteral</code>	Specifies the database literal name for services to be used as per the <code><dbConfig.secretName></code> .	Yes	mysql-username	
<code>&dbCommonConfig.dbPasswordLiteral</code>	Specifies the database literal password for services to be used as per the <code><dbConfig.secretName></code> .	Yes	mysql-password	

Common Configurations Service and Database configurations in nrf-client-nfdiscovery

Table 3-62 Common Configurations Service and Database configurations in nrf-client-nfdiscovery

Parameter	Description	Mandatory Parameter	Default Value	Notes
nrf-client-nfdiscovery.commonCfg Server.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfdiscovery.dbConfig.db Host	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfdiscovery.dbConfig.db Port	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfdiscovery..dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-db-pass	Same value as global.dbCredSecretName
nrf-client-nfdiscovery.dbConfig.db Name	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfdiscovery.dbConfig.db UNameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfdiscovery.dbConfig.db PwdLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Common Configurations Service and Database configurations in nrf-client-nfmanagement**Table 3-63 Common Configuration Service and Database configurations in nrf-client-nfmanagement**

Parameter	Description	Mandatory Parameter	Default Value	Notes
nrf-client-nfmanagement.common CfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfmanagement.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfmanagement.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfmanagement.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-privilege-d-db-pass	Same value as global.privilegedDbCredSecretName
nrf-client-nfmanagement.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfmanagement.dbConfig.dbUNameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
nrf-client-nfmanagement.dbConfig.dbPwdLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Common Configurations Service and Database configurations in appinfo

Table 3-64 Common Configuration Service and Database configurations in appinfo

Parameter	Description	Mandatory Parameter	Default Value	Notes
appinfo.commonCfgClient.enabled	Specifies whether to enable or disable common config client for common config service.	Yes	true	
appinfo.commonCfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
appinfo.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
appinfo.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
appinfo.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-db-pass	Same value as global.dbCredSecretName
appinfo.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
appinfo.dbConfig.dbUNameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Table 3-64 (Cont.) Common Configuration Service and Database configurations in appinfo

Parameter	Description	Mandatory Parameter	Default Value	Notes
appinfo.dbConfig.dbPwd Literal	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Non real-time based status API from the monitor service is dependent on the Prometheus. If Prometheus-server and prometheus-kube-state-metrics is not working or installed properly then the non real-time API provides the wrong value.

It is recommended to use real-time DBstatus URIs because these URIs always provide the right values.

For example:

```
db_status_uri : http://occnDbTier-db-monitor-svc:8080/db-tier/status/cluster/local/realtime
realtime_db_status_uri : http://occnDbTier-db-monitor-svc:8080/db-tier/status/cluster/local/realtime
replication_status_uri : http://occnDbTier-db-monitor-svc:8080/db-tier/status/replication/realtime
```

Common Configuration Service and Database configurations in perf-info

Table 3-65 Common Configuration Service and Database configurations in perf-info

Parameter	Description	Mandatory Parameter	Default Value	Notes
perf-info.commonCfgClient.enabled	Specifies whether to enable or disable common config client for common config service.	Yes	true	
perf-info.commonCfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
perf-info.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Table 3-65 (Cont.) Common Configuration Service and Database configurations in perf-info

Parameter	Description	Mandatory Parameter	Default Value	Notes
perf-info.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
perf-info.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	occpn-db-pass	Same value as global.dbCredSecretName
perf-info.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
perf-info.dbConfig.dbUName Literal	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
perf-info.dbConfig.dbPwdLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Common Configuration Service and Database configurations in ingress-gateway**Table 3-66 Common Configuration Service and Database configurations in ingress-gateway**

Parameter	Description	Mandatory Parameter	Default Value	Notes
ingress-gateway.commonCfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Table 3-66 (Cont.) Common Configuration Service and Database configurations in ingress-gateway

Parameter	Description	Mandatory Parameter	Default Value	Notes
ingress-gateway.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
ingress-gateway.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
ingress-gateway.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-db-pass	Same value as global.dbCredSecretName
ingress-gateway.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
ingress-gateway.dbConfig.dbUNameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
ingress-gateway.dbConfig.dbPwdLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Common Configuration Service and Database configurations in egress-gateway

Table 3-67 Common Configuration Service and Database configurations in egress-gateway

Parameter	Description	Mandatory Parameter	Default Value	Notes
egress-gateway.commonCfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
egress-gateway.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
egress-gateway.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
egress-gateway.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-db-pass	Same value as global.dbCredSecretName
egress-gateway.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
egress-gateway.dbConfig.dbUsernameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
egress-gateway.dbConfig.dbPasswordLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Common Configuration Service and Database configurations in alternate-route

Table 3-68 Customizable Parameters for Common Configuration Service in alternate-route

Parameter	Description	Mandatory Parameter	Default Value	Notes
alternate-route.commonCfgServer.port	Specifies the common config server port for common config service.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
alternate-route.dbConfig.dbHost	Specifies the MySQL database host for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
alternate-route.dbConfig.dbPort	Specifies MySQL database port for services.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
alternate-route.dbConfig.secretName	Specifies kubernetes secret object name from which MYSQL username and password is picked.	Yes	ocnp-db-pass	Same value as global.dbCredSecretName
alternate-route.dbConfig.dbName	Specifies common config database name for services to store common configurations.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
alternate-route.dbConfig.dbUNameLiteral	Specifies the database literal name for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.
alternate-route.dbConfig.dbPwdLiteral	Specifies the database literal password for services to be used as per the <dbConfig.secretName>.	Yes	Same as the value provided in the Table 3-61	To use a different values than the default value, remove the comment (#) from the respective parameters and edit the values.

Note

You can add additional parameters under the dbConfig for each service by adding key value pair after the <<: *dbCommonConfig text. The following snippet shows an example:

```
dbConfig:  
  <<: *dbCommonConfig  
  <key>:<value>
```

where, <key> is the parameter to be configured and <value> is the configured value for <key>.

3.33 Graceful Shutdown Configurations

This section describes the customizations that can be done in ocbsf_custom_values_26.1.201.yaml file to configure graceful shutdown of Kubernetes pods.

Table 3-69 Configurable Parameters for Graceful Termination in BSF services

Parameter	Description	Mandatory/ Optional Parameter	Default Value
- bsf-management-service.gracefulShutdown.gracePeriod - config-server.gracefulShutdown.gracePeriod - cm-service.gracefulShutdown.gracePeriod - queryservice.gracefulShutdown.gracePeriod - audit-service.gracefulShutdown.gracePeriod - nrf-client.nrf-client-nfdiscovery.gracefulShutdown.gracePeriod - nrf-client.nrf-client-nfmanagement.gracefulShutdown.gracePeriod - appinfo.gracefulShutdown.gracePeriod - perf-info.gracefulShutdown.gracePeriod - diam-gateway.gracefulShutdown.gracePeriod - ingress-gateway.gracefulShutdown.gracePeriod - egress-gateway.gracefulShutdown.gracePeriod	Specifies the waiting grace period for current requests to be processed. If there are no current requests then this period is neglected.	Optional	30s Note: 's' in case of seconds and 'm' in case of minutes.

Here is a sample configuration for graceful shutdown parameters in ocbf_custom_values_26.1.201.yaml file:

```
bsf-management-service:
  # Graceful Termination
  gracefulShutdown:
    gracePeriod: 30s

config-server:
  # Graceful Termination
  gracefulShutdown:
    gracePeriod: 30s

cm-service:
  # Graceful Termination
```

```
gracefulShutdown:
  gracePeriod: 30s
```

3.34 Configurations for NodeSelector

Kubernetes nodeSelector feature is used for manual pod scheduling. A pod is assigned to only those nodes that have label(s) identical to label(s) defined in the nodeSelector.

To see all the labels attached to a node you can run:

```
kubectl describe node pollux-k8s-node-1
Name:                pollux-k8s-node-1
Roles:               <none>
Labels:              beta.kubernetes.io/arch=amd64
                    kubernetes.io/hostname=pollux-k8s-node-1
                    kubernetes.io/os=linux
                    topology.kubernetes.io/region=RegionOne
                    topology.kubernetes.io/zone=nova
```

The default labels attached to kubernetes nodes are displayed. In order to assign a pod to the node in BSF, you need to set custom configurations in ocsbf_custom_values_26.1.201.yaml file.

You can configure nodeselection field under global/local services section of the ocsbf_custom_values_26.1.201.yaml file. For ingress gateway, egress gateway and alternate route services nodeSelector is configured at global section.

Table 3-70 Configurations for NodeSelector

Parameter	Description	Values	Notes
global.nodeSelection	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - ENABLED - DISABLED Default Value: DISABLED	global: nodeSelection: ENABLED
global.nodeSelector.nodeKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelector: nodeKey: key nodeValue: value For example: global: nodeSelection: ENABLED nodeSelector: nodeKey: 'kubernetes.io/os'

Table 3-70 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
global.nodeSelector.nodeValue	Specifies valid value pair for the above key for a label for a particular node.	'Not Applicable'	nodeValue: 'linux'

Table 3-71 Configurations for NodeSelector

Parameter	Description	Values	Notes
bsf-management-service.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: • true • false Default Value: false	bsf-management-service:
bsf-management-service.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true
bsf-management-service.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorKey: key nodeSelectorValue: value For example: bsf-management-service: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux
config-server.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: • true • false Default Value: false	config-server:
config-server.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true nodeSelectorKey: key

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
config-server.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorValue : value For example: <pre>config-server: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux</pre>
queryservice.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	queryservice:
queryservice.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true nodeSelectorKey: key
queryservice.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorValue : value For example: <pre>queryservice: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux</pre>

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
cm-service.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	cm-service: nodeSelectorEnabled: true nodeSelectorKey: key nodeSelectorValue: value For example: cm-service: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux
cm-service.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	
cm-service.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	
audit-service.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	audit-service: nodeSelectorEnabled: true nodeSelectorKey: key nodeSelectorValue: value For example: audit-service: nodeSelectorEnabled: true
audit-service.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
audit-service.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux
nrf-client.nrf-client-nfdiscovery.global.deploymentNrfClientService.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: • true • false Default Value: false	nrf-client: nrf-client-nfdiscovery: global:
nrf-client.nrf-client-nfdiscovery.global.deploymentNrfClientService.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	ephemeralStorageLimit: 1024 deploymentNrfClientService: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux For example: nrf-client: nrf-client-nfdiscovery: global: ephemeralStorageLimit: 1024 deploymentNrfClientService: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
nrf-client.nrf-client-nfdiscovery.global.deploymentNrfClientService.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorValue : linux
nrf-client.nrf-client-nfmanagement.global.deploymentNrfClientService.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	nrf-client: nrf-client-nfmanagement: global: deploymentNrfClientService: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue : linux For example: nrf-client: nrf-client-nfmanagement: global: deploymentNrfClientService: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue : linux
nrf-client.nrf-client-nfmanagement.global.deploymentNrfClientService.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	
nrf-client.nrf-client-nfmanagement.global.deploymentNrfClientService.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	
appinfo.nodeSelection	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - ENABLED - DISABLED Default Value: DISABLED	appinfo: nodeSelection:

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
appinfo.nodeSelector	Specifies the key value pair for a label of a particular node.	Not Applicable	ENABLED nodeSelector: key: value For example: <pre>appinfo: nodeSelector: ENABLED nodeSelector: 'kubernetes.io/os': 'linux'</pre>
perf-info.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	perf-info:
perf-info.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true nodeSelectorKey: key
perf-info.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorValue: value For example: <pre>perf-info: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux</pre>
diam-connector.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: - true - false Default Value: false	diam-connector:

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
diam-connector.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true
diam-connector.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux For example: diam-connector: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux
diam-gateway.nodeSelectorEnabled	Specifies if pods needs to assigned to a specific node manually or not.	Allowed Values: • true • false Default Value: false	diam-gateway:
diam-gateway.nodeSelectorKey	Specifies a valid key that is a node label of a particular node in the cluster.	Not Applicable	nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os nodeSelectorValue: linux For example: diam-gateway: nodeSelectorEnabled: true nodeSelectorKey: kubernetes.io/os

Table 3-71 (Cont.) Configurations for NodeSelector

Parameter	Description	Values	Notes
diam-gateway.nodeSelectorValue	Specifies valid value pair for the above key for a label of a particular node.	Not Applicable	nodeSelectorValue : linux

3.35 Configuring Kafka for NF message feed

This section describes the parameters that are required to configure Kafka for NF message feed.

Table 3-72 Parameters for Message Feed Configuration for Kafka

Parameter	Description	Mandatory/Optional Parameter	Default Value
global.nfType	Identifies the type of producer NF.	Optional	BSF
global.nfInstanceId	Identifies the producer NF instance.	Optional	6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c
global.nfFqdn	Identifies the producer NF fqdn.	Optional	BSF-d5g.oracle.com

3.36 Configuration Parameters for Dual Stack and Single Stack IPv4 or IPv6

Note

- These parameters are also applicable for single stack IPv6.
- **global.isIpvSixSetup**, **ingress-gateway.isIpv6Enabled**, **egress-gateway.isIpv6Enabled**, and **alternate-route.isIpv6Enabled** are deprecated from 25.2.200.

Table 3-73 Configurations for Dual Stack

Parameter	Description	Mandatory Parameter	Default Value	Added/Deprecated/Updated in Release	Notes
global.deploy mentMode	Indicates deployment mode of the services	Yes	ClusterPreferred	Added in Release 25.1.200	Possible values: IPv4, IPv6, IPv4_IPv6, IPv6_IPv4, ClusterPreferred For more information on deployment mode, see the "Support for Dual Stack" section in Oracle Communications Cloud Native Core, Binding Support Function User Guide.
global.egressRoutingMode	Controls the IP selection and connections for Egress/outgoing traffic.	No	None	Added in Release 25.1.200	Possible values: IPv4, IPv6, IPv4_IPv6, IPv6_IPv4, None
global.dgwRoutingMode	Controls the IP address selection for Diameter Gateway connections when DNS resolves a destination to multiple IP address types.	No	IPv4_IPv6	Added in Release 25.1.201	Possible values: IPv6, IPv4, IPv6_IPv4, IPv4_IPv6

Migration from dualStackPreferredDnsResolutionIpAddressType

The `diam-gateway.dualStackPreferredDnsResolutionIpAddressType` parameter is replaced by `global.dgwRoutingMode`. During an upgrade, remove the retired parameter and set `global.dgwRoutingMode` to the value that matches the required IP-family preference. The `diamgateway.envSupportedIpAddressType` parameter is no longer applicable.

Table 3-74 Routing Mode Migration

diam-gateway.dualStackPreferredDnsResolutionIpAddressType	global.dgwRoutingMode
IPv4	IPv4
IPv6	IPv6
IPv4_IPv6	IPv4_IPv6
IPv6_IPv4	IPv6_IPv4
both	Select IPv4_IPv6 or IPv6_IPv4 explicitly, depending on the required preference.

Note

The dual stack feature is not supported on Oracle Cloud Native Environment (CNE) but is supported on third-party CNEs.

4

Upgrading BSF

This chapter provides information about upgrading Oracle Communications Cloud Native Core, Binding Support Function (BSF) deployment to the latest release. It is recommended to perform BSF upgrade in a specific order. For more information about the upgrade order, see *Oracle Communications Cloud Native Core, Solution Upgrade Guide*.

Note

Unless otherwise stated, features should not be enabled during BSF upgrade using Helm.

It is recommended to enable new features only after ensuring that the BSF upgrade is successful and a rollback to an older release will not be considered.

In a multi-site environment, BSF upgrade must be successful in all the sites before enabling the required features.

This can involve two Helm upgrades: one for upgrading BSF software version and then another Helm upgrade to enable the feature after accepting the upgrade.

If there are any issues observed after enabling the features, disable the feature using Helm and CNC Console, perform a Helm upgrade, and verify that the issue is resolved.

Note

- In a georedundant deployment, perform the steps explained in this section on all the georedundant sites.
- For BSF georedundant deployments, during a site upgrade the difference between the BSF release versions for all the georedundant sites cannot be more than 1 release.

For example, In a three-site BSF deployment, all the 3 sites are at the same release version as N. During site upgrade, site 1 and site 2 are upgraded to N+1 version, and site 3 is not upgraded yet. At this State, before upgrading site 3 to N+1, upgrading site 1 and/or site 2 from N+1 version to N+2 version is not supported as the difference between the BSF release versions for all the georedundant sites cannot be more than 1 release.

For more information about the cnDBTier georedundant deployments, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

For more information about the CNC Console georedundant deployments, see *Oracle Communications Cloud Native Core, CNC Console Installation, Upgrade, and Fault Recovery Guide*.

4.1 Supported Upgrade Paths

The following table lists the supported upgrade paths for BSF:

Source Release	Target Release
25.2.2xx	26.1.201

Note

BSF must be upgraded before upgrading cnDBTier.

4.2 Upgrade Strategy

BSF supports in-service upgrade. The supported upgrade strategy is RollingUpdate. The rolling update strategy is a gradual process that allows you to update your Kubernetes system with only a minor effect on performance and no downtime. The advantage of the rolling update strategy is that the update is applied Pod-by-Pod so the greater system can remain active.

The following engineering configuration parameters are used to define upgrade strategy:

- `updateStrategy` parameter indicates the update strategy used in BSF.
- `maxUnavailable` parameter determines the maximum number of pods that can be unavailable during upgrade.
For more information on `maxUnavailable` for each microservices refer [PodDisruptionBudget Configuration](#) section.

Note

When BSF is deployed with OCCM, follow the specific upgrade sequence as mentioned in the *Oracle Communications, Cloud Native Core Solution Upgrade Guide*.

4.3 Preupgrade Tasks

This section provides information about preupgrade tasks to be performed before upgrading BSF.

1. Backup the current `custom_values.yaml` file.
2. Update the new `custom_values.yaml` file for target BSF release. For details on customizing this file, see [Customizing BSF](#).
3. Before starting the upgrade, take a manual backup of BSF REST based configuration. This helps if preupgrade data has to be restored.

Note

For Rest API configuration details, see *Oracle Communications Cloud Native Core, Binding Support Function REST Specification Guide*.

4. To add an index at site level, configure the value of `performAddDropIndexAtSiteLevel` parameter to `TRUE` in `custom_values.yaml` file. For more details on this parameter, see [Configurations for Pre and Post Upgrade/Install Validations](#). Provide the following privileges to `bsfprivilegedusr`:

```
$ mysql -h 127.0.0.1 -u<root user> -p<root password>
```

```
mysql> GRANT SYSTEM_VARIABLES_ADMIN ON *.* TO 'bsfprivilegedusr'@'%' with  
GRANT option;
```

5. Before upgrading, perform sanity check using Helm test. See the [Performing Helm Test](#) section for the Helm test procedure.
6. Install or upgrade the network policies, if applicable. For more information, see [Configuring Network Policies](#).

Note

It is recommended to perform BSF upgrade in a specific order. For more information about the upgrade order, see *Oracle Communications Cloud Native Core, Solution Upgrade Guide*.

Before starting the procedure to upgrade Binding Support Function (BSF), perform the following tasks:

- Since Diameter Connector is removed from BSF, perform any of the following methods to prepare system for upgrade with functional Diameter Connector:
One-step Upgrade: Set the value of `diamConnectorEnable` as `false` in the custom values yaml file and then perform the upgrade. This method is a site isolation upgrade and may have an impact on traffic.

OR

Two-step Upgrade:

1. Set the value of the `diamConnectorEnable` parameter as **true** in the custom values yaml file and perform the upgrade.
2. After a successful upgrade, change the value of `diamConnectorEnable` to **false** in the same custom values yaml file and rerun the upgrade command.
Running the same upgrade command twice by changing the `diamConnectorEnable` parameter values does not impact the traffic.

4.4 Upgrade Tasks

This section includes information about upgrading an existing Binding Support Function (BSF) deployment.

Helm Upgrade

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

Upgrade Procedure

⚠ Caution

- Stop the provisioning traffic before you start the upgrade procedure.
- Do not perform any configuration changes during the upgrade.
- Do not exit from `helm upgrade` command manually. After running the `helm upgrade` command, it takes some time (depending upon the number of pods to upgrade) to upgrade all the services. In the meantime, you must not press "ctrl+c" to come out from `helm upgrade` command. It may lead to anomalous behavior.

📘 Note

While upgrading from BSF 25.2.201 or later 25.2.2xx releases to 26.1.201, you are not required to perform a double upgrade, as the duplicate port issue is resolved starting from these versions. However, while upgrading from BSF 25.2.200 or any earlier version to 26.1.201, you should run the Helm upgrade twice in order to fully eliminate a duplicate container port definition condition (that is, duplicate port 9000). The second upgrade must run with the Helm flag `--no-hooks`.

1. Unzip the release package file to the system where you want to deploy BSF. The BSF release package is named using the following convention:

`ReleaseName-pkg-ReleaseNumber.tgz`

where, `ReleaseName` is the name used to track a particular installation instance and

`ReleaseNumber` is the release number.

For BSF 26.1.201, the release package file name is `ocbsf-pkg26.1.201.0.0.tgz`.

📘 Note

For information on how to download the package from [My Oracle Support \(MOS\)](#), see [Downloading BSF Package](#).

2. Untar the BSF package file to get the docker image tar file:

```
tar -xvzf ReleaseName-pkg-ReleaseNumber.tgz
```

For Example:

```
tar -xvzf ocbsf-pkg-26.1.201.0.0.tgz
```

The directory consists of the following:

- ocbsf-images-26.1.201.tar - BSF Docker Images File
 - ocbsf-26.1.201.tgz - Helm Charts
 - Readme.txt - Readme txt file
 - ocbsf-26.1.201.tgz.sha256 - Checksum for Helm chart tgz file
 - ocbsf-images-26.1.201.tar.sha256 - Checksum for images tgz file
3. Load and Push the Images to Customer Docker Registry. See [Pushing the Images to Customer Docker Registry](#).
 4. Modify the required custom-values.yaml file for the target BSF release. To learn how to customize the file or any specific parameters, see [Customizing BSF](#).

Note

The values of the parameters mentioned in the custom values yaml file overrides the defaults values specified in the helm chart. If the **envMySQLDatabase** parameter is modified, then you should modify the **configDbName** parameter with the same value.

5. Upgrade BSF using Helm:

```
helm upgrade <release-name> <helm-chart> -f <custom-file> -n <release-namespace>
```

Example:

```
helm upgrade ocbsf ocbsf-26.1.201.tgz -f ocbsf-26.1.201-custom-values.yaml  
-n ocbsf
```

where:

helm_chart is the location of the helm chart extracted from ocbsf-26.1.201.tgz file

release_name is the release name used by helm command.

release_namespace is the deployment namespace used by helm command.

custom_file - is the name of the custom values yaml file.

Optional parameters that can be used in the `helm install` command:

- **atomic**: If this parameter is set, installation process purges chart on failure. The `--wait` flag will be set automatically.
- **wait**: If this parameter is set, installation process will wait until all pods, PVCs, Services, and minimum number of pods of a deployment, StatefulSet, or ReplicaSet are in a ready state before marking the release as successful. It will wait for as long as `--timeout`.
- **timeout duration**: If not specified, default value will be 300 (300 seconds) in Helm. It specifies the time to wait for any individual Kubernetes operation (like Jobs for hooks).

If the `helm install` command fails at any point to create a Kubernetes object, it will internally call the purge to delete after the timeout value. Here, the timeout value is not for overall installation, but for automatic purge on installation failure.

Note

It is recommended not to use `--wait` and `--atomic` parameters along with `helm upgrade` as this might result in upgrade failure.

6. Run `helm upgrade` again with `--no-hooks` to upgrade an existing BSF deployment:

```
helm upgrade <release-name> <helm_chart> -f <occnp_customized_values.yaml>
--namespace <namespace> --no-hooks
```

7. Run the following command to check the status of jobs and pods:

```
kubectl get jobs,pods -n release_namespace
```

For example:

```
kubectl get jobs,pods -n ocbsf
```

You should see the status as **Running** for all the pods if BSF is deployed successfully.

8. Before upgrading, perform sanity check using Helm test. See the [Performing Helm Test](#) section for the Helm test procedure.
9. If the upgrade for BSF fails due to any of the following conflicts, you are required to delete the conflicting items using the relevant commands as follows:
 - In case of conflict due to rolebinding, run the following command:

```
kubectl delete rolebinding <conflicting item> -n <namespace>
```
 - In case of conflict due to PodDisruptionBudget, run the following command:

```
kubectl delete PodDisruptionBudget <conflicting item> -n <namespace>
```

If the upgrade fails due to any other reason, see *Upgrade or Rollback Failure in Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*.

Note

The following warnings must be ignored for BSF upgrade on CNE 25.1.2xx and 25.2.1xx:

```
helm upgrade <release-name> -f <custom.yaml> <tgz-file> -n <namespace>
W0301 07:39:30.096125 1718193 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port definition
with spec.template.spec.containers[0].ports[1]
W0301 07:39:34.033420 1718193 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port definition
with spec.template.spec.containers[0].ports[2]
Release "<release-name>" has been upgraded. Happy Helming!
NAME: <release-name>
LAST DEPLOYED: <Date-Time>
NAMESPACE: <namespace>
STATUS: deployed
REVISION: <N>
```

Note

To automate the lifecycle management of the certificates through OCCM, you can migrate certificates and keys from BSF to OCCM. For more information, see "Introducing OCCM in an Existing NF Deployment" in Oracle Communications Cloud Native Core, Certificate Management User Guide.

You can remove Kubernetes secrets if the current version of BSF does not use that secret by checking the `ocbsf_custom_values.yaml` file. Before deleting, please make sure that there is no plan to rollback to the BSF version which uses these secrets. Otherwise Rollback will fail.

4.5 Postupgrade Tasks

This section explains the postupgrade tasks for BSF.

4.5.1 Alert Configuration

This section describes how to modify or update BSF alerts as needed, based on requirements, after performing the upgrade. For more details, see [Configuring BSF Alerts](#) section.

4.6 MIB Management

toplevel.mib and *BSF-ALARM-MIB.mib* are two MIB files which are used to generate the traps. You must update these files along with the Alert file in order to fetch the traps in their environment. The MIB files are managed by SNMP manager.

Note

bsf_alarm_mib.mib file has been replaced by *BSF-ALARM-MIB.mib* file.

5

Rolling Back BSF

This chapter provides information about rolling back Oracle Communications Cloud Native Core, Binding Support Function (BSF) deployment to the previous release. It is recommended to perform BSF rollback in a specific order. For more information about the rollback order, see *Oracle Communications Cloud Native Core Solution Upgrade Guide*.

5.1 Supported Rollback Paths

The following table lists the supported rollback paths for BSF:

Source Release	Target Release
26.1.201	25.2.2xx

If georedundancy feature was disabled before upgrading to 26.1.201, then rolling back to a previous version will automatically disable this feature. However, the database will still have records of the NfInstances and NfSubscriptions from the mated sites. For more information, contact My Oracle Support.

5.2 Rollback Tasks

Note

Ensure that no pod is in the failed state.

To roll back from BSF 26.1.201 to a previous version, perform the following steps:

1. To check which revision you need to roll back to, run any of the following commands:
Using Helm:

```
helm history <release_name> -n <release_namespace>
```

2. Run either of the following commands to roll back to the required version:

- a. Using Helm:

```
helm rollback <release_name> <revision_number> -n <release_namespace>
```

Note

The following warnings must be ignored for BSF rollback to CNE 25.1.2xx and 25.2.1xx:

```
helm rollback <release_name> <revision-version> -n <namespace>
W0910 07:39:23.637460 3540122 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port
definition with spec.template.spec.containers[0].ports[2]
W0910 07:39:26.437684 3540122 warnings.go:70]
spec.template.spec.containers[0].ports[3]: duplicate port
definition with spec.template.spec.containers[0].ports[1]
Rollback was a success! Happy Helming!
```

3. **<Applicable for release 22.2.0 and later >** If you are rolling back from BSF 22.2.0 and later release to 22.1.0 or earlier releases, then run the following command to delete the database that is not applicable:

```
DROP DATABASE ocbsf_LeaderPodDb;
```

Note

If the rollback is not successful, perform the troubleshooting steps mentioned in *Oracle Communications Cloud Native Core, Binding Support Function Troubleshooting Guide*.

Note

When rolling back from 23.4.0, Audit Service must be disabled and re-enabled after the rollback is completed. As the audit registration request is resent after the Audit Service pod comes up and the data gets updated in Audit Service tables.

6

Uninstalling BSF

This chapter provides information about uninstalling Oracle Communications Cloud Native Core, Binding Support Function (BSF).

When you uninstall a Helm chart from BSF deployment, it removes only the Kubernetes objects that were created during the installation.

6.1 Uninstalling CNC BSF using Helm

This chapter describes how to uninstall BSF using Helm.

To uninstall BSF, run the following command:

For Helm 2:

```
helm uninstall -n <release_name>
```

Here, `release_name` is a name provided by the user to identify the helm deployment.

`release-namespace` is a name provided by the user to identify the namespace of BSF deployment.

Helm keeps a record of its releases, so you can still reactivate the release after uninstalling it.

To completely remove a release from the cluster, add the `--purge` parameter to Helm 2 uninstall command:

```
helm delete --purge release_name
```

For example, to completely remove a release named "ocbsf", run the following command:

```
helm delete --purge ocbsf
```

① Note

- When you uninstall both BSF and `cnDBTier` network functions, it is recommended to delete PVC (PersistentVolumeClaims) volumes of `cnDBTier` that were created at the time of installing `cnDBTier`. For more information on how to delete PVC volumes, see [Deleting PVC Volumes](#).
- If you are uninstalling only BSF network function, do not delete PVC volumes.

6.2 Deleting Kubernetes Namespace

This chapter describes how to delete Kubernetes namespace where BSF is deployed.

To delete the Kubernetes namespace, run the following command:

```
kubectl delete namespace <release-namespace>
```

where *release_namespace* is the deployment namespace used by the helm command.

For example, to delete a kubernetes namespace named "ocbsf", run the following command:

```
kubectl delete namespace ocbsf
```

6.3 Removing MySQL Users

This chapter describes how to remove MySQL users.

To remove MySQL users while uninstalling BSF, run the following commands:

```
SELECT user FROM mysql.user;
DROP USER 'bsfprivilegedusr'@'%';
DROP USER 'bsfusr'@'%';
```

6.4 Deleting PVC Volumes

This section describes how to delete a PVC volume.

Following is the procedure for to delete a PVC volume:

1. Get a list of the PVC volumes for the required cnDBTier namespace by running the following command:

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

where, <namespace> is the namespace of Policy deployment.

For example:

```
kubectl get pvc -n occnp
```

Sample Output

NAME			STATUS		CAPACITY	ACCESS MODES
VOLUME						
STORAGECLASS	AGE					
pvc-backup-ndbmt-d-0			Bound		pvc-56420da4-f70c-46fd-8f0a-	
c33ff50ddd98	3Gi	RWO			standard	26d
pvc-backup-ndbmt-d-1			Bound		pvc-d7e3ef21-4161-40cc-	
abc9-756dfd3c0674	3Gi	RWO			standard	26d
pvc-ndbappmysqld-ndbappmysqld-0			Bound		pvc-d76f0cbd-	
c0e5-48f1-9a83-1353dbee6c41	2Gi			RWO	standard	26d
pvc-ndbappmysqld-ndbappmysqld-1			Bound		pvc-c31749df-ae98-48c1-	
a8a7-449535d0f60a	2Gi	RWO			standard	26d
pvc-ndbmgmd-ndbmgmd-0			Bound		pvc-765a9a4b-726d-43fe-af91-	
a3a6c85ab321	1Gi	RWO			standard	26d
pvc-ndbmgmd-ndbmgmd-1			Bound		pvc-71b5877e-a753-4fac-	

b995-5fdf2d465af8	1Gi	RWO	standard	26d
pvc-ndbmt-d-ndbmt-d-0		Bound	pvc-e0c5f263-d5d3-4f18-	
bb11-44ceaa6a2305	3Gi	RWO	standard	26d
pvc-ndbmt-d-ndbmt-d-1		Bound	pvc-d03f799d-abe2-4b71-96c4-	
c98cacdbeaba	3Gi	RWO	standard	26d
pvc-ndbmysqld-ndbmysqld-0		Bound	pvc-	
d80321a9-5239-4e34-9bcf-11053c9de5ef	2Gi	RWO		
standard	26d			
pvc-ndbmysqld-ndbmysqld-1		Bound	pvc-545e0f35-	
ad20-4c24-927a-6f1e49b06cc9	2Gi	RWO	standard	26d

2. Delete all the PVC volumes in the cnDBTier namespace by running the following command:

```
kubectl -n <namespace> delete pvc <pvc_name>
```

Example:

```
kubectl -n p1 delete pvc pvc-backup-ndbmt-d-ndbmt-d-0
kubectl -n p1 delete pvc pvc-backup-ndbmt-d-ndbmt-d-1
kubectl -n p1 delete pvc pvc-ndbappmysqld-ndbappmysqld-0
kubectl -n p1 delete pvc pvc-ndbappmysqld-ndbappmysqld-1
kubectl -n p1 delete pvc pvc-ndbm-gmd-ndbm-gmd-0
kubectl -n p1 delete pvc pvc-ndbm-gmd-ndbm-gmd-1
kubectl -n p1 delete pvc pvc-ndbmt-d-ndbmt-d-0
kubectl -n p1 delete pvc pvc-ndbmt-d-ndbmt-d-1
kubectl -n p1 delete pvc pvc-ndbmysqld-ndbmysqld-0
kubectl -n p1 delete pvc pvc-ndbmysqld-ndbmysqld-1
```

6.5 Uninstalling Site in Georedundant Deployment

This chapter describes how to uninstall a site (except the last site) when BSF is deployed in georedundant setup.

Deleting entries of unique databases

For georedundant deployment, run the following command to delete entries of unique databases from the occnp_release database:

```
delete from ReleaseConfig where SiteId='7c4f7f05-ffdd-408f-ba78-b2c4dc83b1fd'
AND CfgKey='public.hook.audit-service';
delete from ReleaseConfig where SiteId='7c4f7f05-ffdd-408f-ba78-b2c4dc83b1fd'
AND CfgKey='public.hook.cms-service';
delete from ReleaseConfig where SiteId='7c4f7f05-ffdd-408f-ba78-b2c4dc83b1fd'
AND CfgKey='public.hook.config-server';
```

Cleaning up NDB Replication Table

In addition, clean up the entries in "mysql.ndb_replication" table by running the following command:

1. Select cast(db as char), cast(table_name as char), cast(conflict_fn as char) from mysql.ndb_replication:

```
mysql> select cast(db as char), cast(table_name as char), cast(conflict_fn
as char) from mysql.ndb_replication;
```

Sample Output

```
+-----+-----+
+-----+
| cast(db as char) | cast(table_name as char) | cast(conflict_fn as
char) |
+-----+-----+
+-----+
| ocpm_bsf        | pcf_binding              |
NDB$MAX_DELETE_WIN(last_access_timestamp) |
+-----+-----+
+-----+
1 rows in set (0.01 sec)
```

If the output for this command shows BSF databases as shown in the sample output, then perform Step 2.

2. Delete from mysql.ndb_replication:

```
delete from mysql.ndb_replication where cast(db as char)("<database_name>");
```

Example

```
delete from mysql.ndb_replication where cast(db as char)="ocpm_bsf";
```

6.6 Uninstalling Last Site in Georedundant Deployment

This chapter describes how to uninstall the last site when BSF is deployed in georedundant setup. The same steps can be performed to uninstall BSF in standalone deployment.

6.6.1 Cleaning up NDB Replication Table

This chapter describes how to cleanup NDB replication table while uninstalling BSF.

To clean up the entries in "mysql.ndb_replication" table, run the following commands:

1. Select cast(db as char), cast(table_name as char), cast(conflict_fn as char) from mysql.ndb_replication:

```
mysql> select cast(db as char), cast(table_name as char), cast(conflict_fn
as char) from mysql.ndb_replication;
```

Sample Output

```
+-----+-----+
+-----+
| cast(db as char) | cast(table_name as char) | cast(conflict_fn as
```

```

char)          |
+-----+-----+
+-----+-----+
| ocpm_bsf      | pcf_binding      |
NDB$MAX_DELETE_WIN(last_access_timestamp) |
+-----+-----+
+-----+-----+
1 rows in set (0.01 sec)

```

If the output for this command shows BSF databases as shown in the sample output, then perform Step 2.

2. Delete from mysql.ndb_replication:

```
delete from mysql.ndb_replication where cast(db as char)("<database_name>");
```

Example

```
delete from mysql.ndb_replication where cast(db as char)="ocpm_bsf";
```

6.6.2 Cleaning up Databases

This chapter describes how to clean up databases when uninstalling BSF.

Note

For georedundant deployment, run the commands provided in this section only if the site being uninstalled is the last site in the complete georedundant group.

To clean up database for the different microservices, run the following command:

```

DROP DATABASE IF EXISTS ocbsf_release;
DROP DATABASE IF EXISTS ocbsf_commonconfig;
DROP DATABASE IF EXISTS ocbsf_config_server;
DROP DATABASE IF EXISTS ocpm_bsf;
DROP DATABASE IF EXISTS ocbsf_audit_service;
DROP DATABASE IF EXISTS ocbsf_cmsservice;
DROP DATABASE IF EXISTS ocbsf_nrf_client;
DROP DATABASE IF EXISTS ocbsf_leaderPodDb;

```

7

Fault Recovery

This chapter provides information about fault recovery for Oracle Communications Cloud Native Core, Binding Support Function (BSF) deployment.

7.1 Overview

You must take database backup and restore it either on the same or a different cluster. It uses the BSF database (MySQL NDB Cluster) is used to run any command or follow instructions.

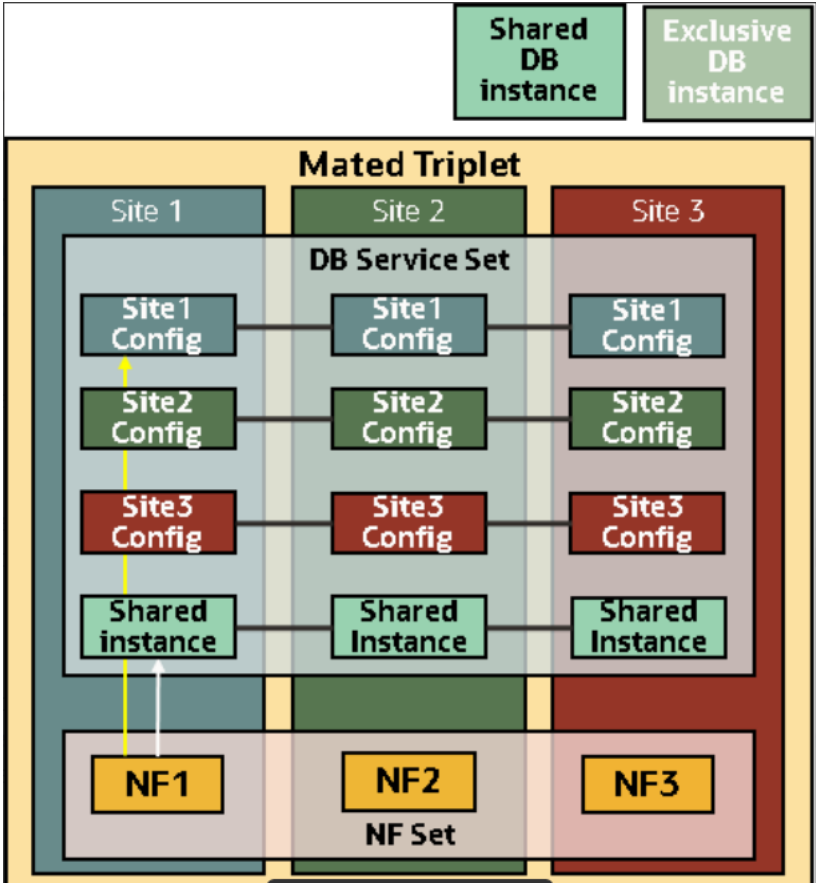
Database Model of BSF

BSF database consists of the following two data types:

- **Configuration Data:** The configuration data is exclusive for a given site. Thus, an exclusive logical database is created and used by a site to store its configuration data. Using CNC Console and Configuration Management service, an operator can configure data in the respective site only.
- **Binding Data:** The binding data is shared across sites. Thus, a common logical database is created and used by all sites. The data is replicated across sites to preserve and share binding information with mated sites. In case of cross sites messaging or a site failure, shared binding data helps in continuity of service.

The following image shows the BSF database model in three different sites:

Figure 7-1 Database Model



7.2 Impacted Areas

The following table shares information about the impacted areas during BSF fault recovery:

Scenario	Requires Fault Recovery or re-install of CNE?	Requires Fault Recovery or re-install of cnDBTier?	Requires Fault Recovery or re-install of CNC BSF?	Other
Scenario 1A: Recovering Corrupted Binding Database(s)	No	Yes Restoring cnDBTier from older backup is the only way to restore back to restore point.	No Only if cnDBTier credentials are changed.	All sites require Fault Recovery.
Scenario: All or Multiple Site Failure	Yes	Yes	Yes	NA

7.3 Prerequisites

Before performing any fault recovery procedure, ensure that the following prerequisites are met:

- DBTier must be in a healthy state and available on multiple sites along with BSF. To check the cnDBTier status, perform the following steps:
 1. Run the following command to ensure that all the nodes are connected:
`ndb_mgm> show`
 2. Run the following command to check the pod status:
`kubectl get pods -n <namespace>`

If the pod status is Running, cnDBTier is in healthy state.
 3. Run the following command to check if the replication is up:
`mysql> show slave status\G`

In case there is any error, see *Oracle Communications Cloud Native Core, DBTier Installation and Upgrade Guide*.
 4. Run the following command to check which DBTier is having ACTIVE replication to take backup:
`select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;`
- You must enable automatic backup on DBTier. Enabling automatic backup helps in achieving the following:
 - Restore stable version of the network function database
 - Minimize significant loss of data due to upgrade or roll back failures
 - Minimize loss of data due to system failure
 - Minimize loss of data due to data corruption or deletion due to external input
 - Migrate database information for a network function from one site to another
- The following files must be available for fault recovery:
 - Custom values file
 - Helm charts
 - Secrets and Certificates
 - RBAC resources

For details on enabling automatic backup, see *Fault Recovery* section in *Oracle Communications Cloud Native Core, cnDBTier (cnDBTier) Installation, Upgrade and Fault Recovery Guide*.

7.4 Fault Recovery Scenarios

This section describes the fault recovery procedures for various scenarios.

Note

This chapter describes scenario based procedures to restore BSF databases only. To restore all the databases that are part of DBTier, see *Fault Recovery* chapter in *Oracle Communications Cloud Native Core, DBTier Installation, Upgrade and Fault Recovery Guide* available on My Oracle Support (MOS).

7.4.1 Scenario: Binding Database Corruption

This section describes how to recover BSF when its binding database corrupts.

When the binding database corrupts, the database on all other sites can also corrupt due to data replication. It depends on the replication status after the corruption has occurred. If the data replication is broken due to database corruption, DBTier fails in either single or multiple sites (not all sites). And if the data replication is successful, then database corruption replicates to all the DBTier sites and DBTier fails in all sites.

The fault recovery procedure covers following sub-scenarios:

- [When cnDBTier Failed in All Sites](#)

7.4.1.1 When cnDBTier Failed in All Sites

This section describes how to recover binding database when successful data replication corrupts all the cnDBTier sites.

To recover binding database, perform the following steps:

1. Uninstall BSF Helm charts on all sites. For more information about uninstalling Helm charts, see *Oracle Communication Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide* available on [MOS](#).
2. Perform DBTier fault recovery procedure:
 - a. Use auto-data backup file for restore procedure. For more information about cnDBTier restore, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* available on [MOS](#).
3. Install BSF Helm charts. For more information about installing Helm charts, see *Oracle Communication Cloud Native Core, Binding Support Function Installation, Upgrade and Fault Recovery Guide* available on [MOS](#).

Note

You can also refer to the `ocbsf_custom_values_26.1.201.yaml` file used at the time of BSF installation for Helm charts installation.

7.4.2 Scenario: Site Failure

This section describes how to perform fault recovery when either one or many of your sites have software failure. This section consists of the following:

- [Single or Multiple Site Failure](#)

7.4.2.1 Single or Multiple Site Failure

This scenario applies when one or more sites, and not all sites, have failed and there is a requirement to perform fault recovery. It is assumed that the user has cnDBTier and BSF installed on multiple sites with automatic data replication and backup enabled.

Note

It is assumed that one of the cnDBTier is in healthy state.

To recover the failed sites, perform the following steps:

Note

Ensure that all the prerequisites mentioned are met.

1. Uninstall BSF. For more information, see the *Uninstalling BSF* section in *Oracle Communications Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide*.
2. Install a new cluster by performing the Cloud Native Environment (CNE) installation procedure. For more information, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide* available on [My Oracle Support](#).
3. Install cnDBTier, in case replication is down or cnDBTier pods are not up and running. For information about installing cnDBTier, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade and Fault Recovery Guide*.
4. Perform DBTier fault recovery procedure:
 - a. Perform DBTier fault recovery procedure to take backup from older healthy site by following the *Create On-demand Database Backup* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade and Fault Recovery Guide*.
 - b. Restore the database to new site by following the *Restore Database with Backup* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
5. Install BSF Helm charts. For more information on installing Helm charts, see *Oracle Communications Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide*.

A

Resilient BSF Microservices with Kubernetes

BSF microservices pods run on Kubernetes cluster. Occasionally, Kubernetes pod disruptions may occur within the cluster, either from voluntary or involuntary causes. This results in pod/ service failing, and the cluster experiences some outage resulting in discontinuity of the service. In order to mitigate these disruptions, the BSF Kubernetes cluster and the services running on it are designed to be resilient by adapting the recommendations and strategies from Kubernetes framework. Thus, improving the availability, prevent/minimize the downtime and outages before they occur.

Described below are the various failure points that might occur in the BSF cluster, and the resilience model that is adopted to handle it.

Table A-1 Failure Point: Worker Node failure

Recovery	Failure Point: Worker Node failure
Recovery - Multiple pod	By having multiple nodes in a cluster, it provides high availability by scheduling pods in different nodes, removing the single point of failure. By running multiple copies of BSF services/pods reduces the chances of outages and service degradation. For more information about this functionality, see section Resource Requirements to Install BSF Services .
Recovery - Anti-Affinity rules	The placement of the pod and any of its replicas can be controlled using Kubernetes pod affinity and anti-affinity rules. Pod anti-affinity rule is used to instruct Kubernetes not to co-locate pods on the same node. This avoids an outage due to the loss of a node.

Table A-2 Failure Point: Physical Server (Hosting Worker Node/s) failure

Recovery	Failure Point: Physical Server (Hosting Worker Node/s) failure
Recovery - Pod Topology Spread (PTS)	Pod topology spread constraints tells the Kubernetes scheduler how to spread pods across nodes in a cluster. It can be across nodes, zones, regions, or other user-defined topology domains. They allow users to use labels to split nodes into groups. Then users can specify pods using a label selector and indicate to the scheduler how evenly or unevenly those pods can be distributed.

Table A-3 Failure Point: Cluster needs to be upgraded or needs to shut down

Recovery	Failure Point: Cluster needs to be upgraded or needs to shut down
Recovery - PodDisruptionBudget (PDB)	Setting PDB ensures that the cluster have a sufficient number of available replicas, to keep it functioning even during maintenance. Using the PDB, we define a number (or percentage) of pods that can be terminated. With PDB configured, Kubernetes drains a node following the configured disruption schedule. New pods is deployed on other available nodes. This approach ensures Kubernetes schedules workloads in an optimal way while controlling the disruption based on PDB configuration. For more information about this functionality, see section PodDisruptionBudget Configuration.
Recovery - Terminate gracefully	When a pod is evicted, it is gracefully terminated honoring the termination gracePeriod setting in the custom yaml file. For more information about this functionality, see section Graceful Shutdown Configurations.

Table A-4 Failure Point: Pod/Application failure

Recovery	Failure Point: Pod/Application failure
Recovery - Kubernetes Probes	Recovery - Kubernetes Probes Kubernetes provides probes i.e health checks to monitor and act on the state of pods (Containers) and to make sure only healthy pods serve traffic. With help of probes, we can control when a pod should be deemed started, ready for service, or live to serve traffic. Kubernetes gives three types of health checks probes: • Liveness probes let Kubernetes know whether the application is running or not. • Readiness probes let Kubernetes know when the application is ready to serve traffic. • Startup probes let Kubernetes know whether the application has properly started or not.

Table A-5 Failure Point: High traffic Rate

Recovery	Failure Point: High traffic Rate
Recovery - Horizontal Pod Auto-Scaling (HPA)	When there is an increase or drop in the traffic, Kubernetes can automatically increase or decrease the pod replicas that serve the traffic. Horizontal scaling means that the response to increased load is to deploy more pods. If the load decreases, and the number of pods is above the configured minimum, the HorizontalPodAutoscaler instructs the work load resource to scale back down.

Table A-6 NF Communication Failures

Recovery	Any(including intra/inter-NF communication Failures)
Recovery - Metrics/logs	Recovery - All BSF service support metrics/logs to capture the behavior.

BSF Microservices Resilience details

The criticality of the service failures is indicated as HIGH, MEDIUM and LOW, and they mean:

- **HIGH-** Service failure impacts the traffic, and it can not be handled successfully.
- **MEDIUM-** Service failure impacts the traffic, and it cannot be handled successfully by the default processing model.
- **LOW-** Service failure does not impact the traffic directly.

Table A-7 BSF Kubernetes cluster Resiliency details

Service Name	Mu lti-po d	Aff init y/ An ti-affi nit yR ule	HP A	PD B	PT S	No de Sel ect or	Se rvi ce abi lity sta tus tra cki ng	Cri tic alit y	Impact of Service loss/ failure	Overloa d Control/ Protecti on	Depend ent service tracking and reportin g
Alternat e Route Service	Y	Y	0% /80 %	1	N	Y	Y	HI GH	On DNS-SRV based alternate routing is enabled: • Handles subsequent messages on failure of initial producer. • Handles notifications on failure of consumer. • SRV based lookup for NRF and SCP.	N	N

Table A-7 (Cont.) BSF Kubernetes cluster Resiliency details

Service Name	Multipod	Affinity/Anti-affinity Rule	HPA	PDB	PTS	Node Selector	Serviceability status tracking	Criticality	Impact of Service loss/failure	Overload Control/Protection	Dependent service tracking and reporting
App-info	Y	Y	2%/80%	20%	N	Y	Y	HIGH	This service tracks status of all services and cnDBTier. This is used by services like: - Nrf-client for NF registration: On appInfo pod failure, Nrfclient uses the last known state fetched from AppInfo. However, if NrfClient pod also restarts, cache data is lost, NF service will be suspended at NRF. - Diam-gateway to track readiness status of cnDBtier: On appInfo pod failure, diameter-gateway uses the last known state fetched from AppInfo. However, if diameter-gateway pod also restarts, then it will fail to detect DB availability and will not be able to accept signaling traffic.	N	N
Audit Service	N	Y	0%/60%	50%	Y	N	Y	LOW	This service handles stale session cleanup and retry binding create operation. Loss of this service leads to large number of stale records and failure of retry binding sessions.	N	N
BSF Management Service	Y	Y	0%/40%	20%	N	Y	Y	HIGH	--	N	N

Table A-7 (Cont.) BSF Kubernetes cluster Resiliency details

Service Name	Multi-pod	Affinity/Anti-affinity Rule	HP A	PD B	PTS	Node Selector	Serviceability status tracking	Criticality	Impact of Service loss/failure	Overload Control/Protection	Dependent service tracking and reporting
CM Service	Y	Y	-- (fixed set of replicas)	20 %	N	Y	Y	HIGH	This service provides user interface to make configuration changes for policy (including common service configuration e.g. Ingress gateway, Egress gateway, NrfClient etc). If this service fails, other services will not be able to fetch the configuration data. Common services pods can continue to run with existing configurations, but it will get impacted on pod restart cases.	N	N
Config Server	Y	Y	5% / 80 %	20 %	N	Y	Y	HIGH	This service is responsible for providing any configuration change to other services. Other services continue to work with existing configuration data, but container restart or pod scaling will lead to readiness failure, as they can not accept traffic without new configuration.	N	N
Diameter Gateway	Y	Y	-- (fixed set of replicas)	20 %	N	Y	Y	HIGH	The failure of this services impacts all diameter related traffic in a site.	Y Enforce overload control for backend services.	DB status is tracked (through applInfo) using helm configuration, to determine the readiness status of gateway pod.

Table A-7 (Cont.) BSF Kubernetes cluster Resiliency details

Service Name	Multipod	Affinity/Anti-affinity Rule	HPA	PDB	PTS	Node Selector	Serviceability status tracking	Criticality	Impact of Service loss/failure	Overload Control/Protection	Dependent service tracking and reporting
Egress Gateway	Y	Y	0% / 80 %	1	N	Y	Y	HIGH	The loss of this service means NRF marks site as "SUSPENDED" due to loss of HB.	N	None
Ingress Gateway	Y	Y	0% / 80 %	1	N	Y	Y	HIGH	The loss of this service impacts connectivity from SCP, NRF or other peers for ingress flow. Hence it will indicate site failure to peers at network/transport level. Note: There shall not be signaling failure, if consumers perform alternate routing to alternate site to achieve session continuity.	Y Enforce overload control for backend services. Backend as well as IGW rate limiting support.	None
NRF Client NF Discovery	Y	Y	0% / 80 %	25 %	N	Y	Y	MEDIUM	The loss of this service means - On-demand discovery procedures is directly impacted. - Signaling flows are impacted. BSF performs on-demand discovery of UDR, thus failure will lead to missing subscriber profile information. Note: Based on configuration, SM/AM/UE may accept service requests.	N	N

Table A-7 (Cont.) BSF Kubernetes cluster Resiliency details

Service Name	Mu lti- pod	Aff init y/ An ti- affi nit yR ule	HP A	PD B	PT S	No de Sel ect or	Se rvi ce abi lity sta tus tra cki ng	Cri tic ali ty	Impact of Service loss/ failure	Overloa d Control/ Protecti on	Depend ent service tracking and reportin g
NRF Client NF Management Service	Y(P olic y nee d to ena ble mul ti- pod sup por t - cur ren tly set to 1)	Y	0% /80 %	NA	N	Y	Y	HI GH	This service is responsible for NF profile registration with NRF. It also performs NRF HB and NRF health check functionality. Loss of this service for HB Timer interval, means that NRF can mark a given PCF instance as SUSPENDED. As soon as Nrf-mgmt pod becomes available, it will automatically refresh Nf's profile at NRF and bring site back to REGISTERED state (if NRF state was suspended).	None	None
Perf-Info	Y	Y	-- (Fix ed set of repl ica s)	20 %	N	Y	Y	ME DI UM	This service is responsible to calculate load and overload level. This is used by services like: - Nrf-client for load reporting - When perfInfo pod is down, Nrfclient currently use the last known load level fetched from PerfInfo. However, if NrfClient pod also restarts, then it will loose its cache information and hence will report load level as zero. - Ingress gateway/ Diameter-gateway to track overload status of back end services. When perfInfo pod is down, these services currently use the last known state reported by perfInfo.	None	None

Table A-7 (Cont.) BSF Kubernetes cluster Resiliency details

Service Name	Mu lti- po d	Aff init y/ An ti- affi nit yR ule	HP A	PD B	PT S	No de Sel ect or	Se rvi ce abi lity sta tus tra cki ng	Cri tic alit y	Impact of Service loss/ failure	Overloa d Control/ Protecti on	Depend ent service tracking and reportin g
Query Service	Y	Y	0% /80 %	20 %	N	Y	Y	LOW	The loss of this service means, operator will not be able to perform query/session viewers functionality. HA to provide better serviceability.	None	None

B

PodDisruptionBudget Configuration

PodDisruptionBudget (PDB) is a Kubernetes resource that allows to achieve high availability of scalable application services when the cluster administrators perform voluntary disruptions to manage the cluster nodes. PDB restricts the number of pods that are down simultaneously from voluntary disruptions. Defining PDB is helpful to keep the services running undisrupted when a pod is deleted accidentally or deliberately. PDB can be defined for high available and scalable BSF services such as BSF Management Service, Diameter Gateway, app-info, and perf-info.

It allows safe eviction of pods when a Kubernetes node is drained to perform maintenance on the node. It uses the default value of the `maxUnavailable` parameter specified in the Helm chart to determine the maximum number of pods that are unavailable during a voluntary disruption. For example, if `maxPdbUnavailable` is 50%, the evictions are allowed until not more than 50% of the desired replicas are unhealthy.

For more information about this functionality, see <https://kubernetes.io/docs/concepts/workloads/pods/disruptions/#pod-disruption-budgets>.

The following BSF services support PDB:

Table B-1 Default PodDisruptionBudget for BSF

Microservice	PDB Default Value (maxUnavailable)
cm-service	20%
alternate_route	1
ocingress_gateway	1
ocegress_gateway	1
oc-diam-gateway	20%
oc-bsf-management	20%
oc-query	20%
oc-perf-info	20%
nrf-client-discovery	25%
nrf-client-management	NA
app-info	20%
config-server	20%
ocbsf-audit	50%

C

Docker Images

Cloud Native Core Binding Support Function deployment package includes ready-to-use docker images and Helm charts to help you orchestrate containers in Kubernetes.

You can use the Docker images and Helm chart to deploy and manage Pods of BSF product services in Kubernetes. Communication between service pods of services of BSF products are preconfigured in the Helm charts.

The following table lists the docker images for BSF:

Table C-1 Docker Images for BSF

Service Name	Docker Image Name
Alternate Route Service	alternate_route
Audit Service	oc-audit
BSF Service	oc-bsf-management
Diameter Gateway	oc-diam-gateway
NF Test	nf-test
Query Service	oc-query
NRF Client Service	nrf-client
CM Service	oc-config-mgmt
CM Service (for Micronaut only)	cnc-common-config-hook
CM Service	common_config_hook
Debug Tool	ocdebug-tools
Config Server Service	oc-config-server
Performance Monitoring Service	oc-perf-info
Application Info service	app_info
Ingress Gateway	ocingress_gateway
Egress Gateway	ocegress_gateway
Ingress/Egress Gateway update configuration	configurationupdate
Ingress/Egress Gateway init configuration	configurationinit

D

Deployment Service Type Selection

Table D-1 Deployment Service Type Selection

Service Type	Description
ClusterIP	Exposes the service on a cluster-internal IP. Specifying this value makes the service only reachable from within the cluster. This is the default ServiceType.
NodePort	Exposes the service on each worker node's IP (public IP address) at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. You'll be able to contact the NodePort service, from outside the cluster, by requesting <i>NodeIP:NodePort</i> . Most BSF services use NodePort to deploy in this release.
LoadBalancer	Exposes the service externally using a cloud provider's load balancer. NodePort and ClusterIP services, to which the external load balancer will route, are automatically created. For CM Service, API gateway, Diameter Gateway service, it's recommended to use LoadBalancer type. Given that the CNE already integrated with a load balancer (METALLB, for OCCNE deployed on baremetal).