Cisco Ultra Cloud Core CPC Policy- Configuration and Administration Guide, Release 2026.03

First Published: 2026-07-24

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

About this Guide

Conventions Used

The following tables describe the conventions used throughout this documentation.

Notice Type	Description
Information Note	Provides information about important features or instructions.
Caution	Alerts you of potential damage to a program, device, or system.
Warning	Alerts you of potential personal injury or fatality. May also alert you of potential electrical hazards.

Typeface Conventions	Description
Text represented as a `screen display`	This typeface represents displays that appear on your terminal screen, for example: `Login:`
Text represented as **commands**	This typeface represents commands that you enter, for example: **show ip access-list** This document always gives the full form of a command in lowercase letters. Commands are not case sensitive.

Typeface Conventions	Description
Text represented as a **command** *variable*	This typeface represents a variable that is part of a command, for example: **show card** *slot_number* *slot_number* is a variable representing the desired chassis slot number.
Text represented as menu or sub-menu names	This typeface represents menus and sub-menus that you access within a software application, for example: Click the **File** menu, then click **New**

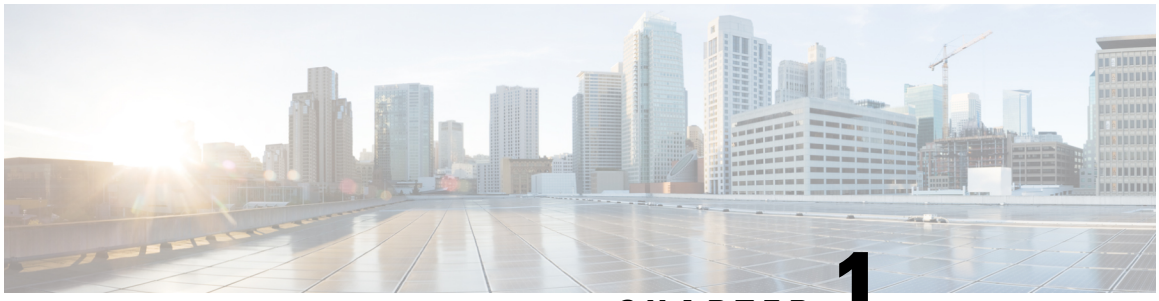
Command Syntax Conventions	Description
{ keyword or *variable* **}**	Required keyword options and variables are those components that are required to be entered as part of the command syntax. Required keyword options and variables are surrounded by grouped braces { }. For example: **sctp-max-data-chunks { limit** *max_chunks* **| mtu-limit }** If a keyword or variable is not enclosed in braces or brackets, it is mandatory. For example: **snmp trap link-status**
[keyword or *variable* **]**	Optional keywords or variables, or those that a user may or may not choose to use, are surrounded by brackets.
|	Some commands support multiple options. These are documented within braces or brackets by separating each option with a vertical bar. These options can be used in conjunction with required or optional keywords or variables. For example: **action activate-flow-detection { intitiation | termination }** or **ip address [count** *number_of_packets* **| size** *number_of_bytes* **]**

Contacting Customer Support

Use the information in this section to contact customer support.

Refer to the support area of http://www.cisco.com for up-to-date product documentation or to submit a service request. A valid username and password are required to access this site. Please contact your Cisco sales or service representative for additional information.

About this Guide



CHAPTER 1

Converged Policy and Charging

CPC overview

Converged Policy and Charging (CPC) is a cloud-native network function (CNF) that provides unified, 3GPP-compliant policy control for both 4G and 5G networks. It integrates subscriber authentication, authorization, accounting (AAA), and policy management into a single, streamlined solution.

Benefits:

- Simplified Migration: CPC eliminates the need for manual migration of user subscriptions or reconfiguration of policy and charging functions (CPC) during the transition from 4G to 5G.
- Reduced Network Complexity: It supports incremental deployment of Standalone (SA) 5G networks by removing the requirement for complex inter-circle mesh connectivity, thus facilitating smoother network evolution.

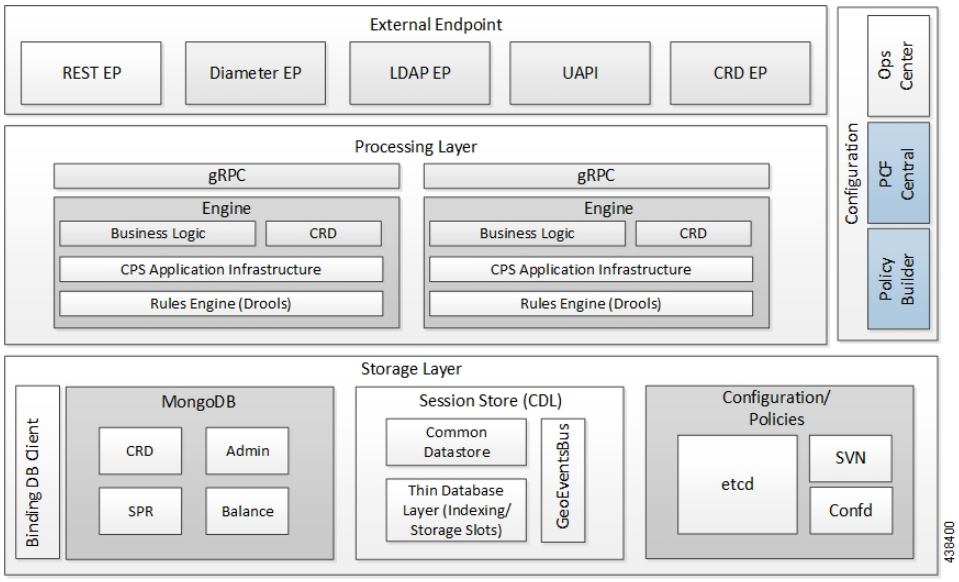
System Composition:

- CPC offers a containerized, cloud-native architecture based on a three-tier microservices framework.
- It supports AAA components using Diameter servers for subscriber management, with protocol, service, and session tiers managing authentication, authorization, accounting, and data persistence.
- The solution includes operational tools such as Policy Builder and Ops-Center for configuration and management, integrated with Cisco Cloud Native Data Plane (CNDP) services for logging, metrics, and alerts.

CPC architecture

The CPC architecture is built on a multi-layer platform, which enables efficient policy control and management in the 5G Core network.

Figure 1: CPC Architecture



At a high level, the components in the architecture perform the following:

1. External Endpoint
   - REST-EP—It is a RESTful interface, which provides a channel for the 5G inbound and outbound messages.
   - LDAP-EP, UAPI, and CRD API—Provides interfaces for CPC communications.
   - Diameter-EP—Responsible for routing the Diameter traffic.
2. Processing Layer
   - grPC—Provides a framework that enables the internal processes to communicate with each other and synchronize their events.
   - CPC-Engine—Hosts the business logic of CPC and responsible for driving the rules engine for making crucial policy decisions.
3. Configurations
   - Policy Builder—Allows configuration of the CPC cluster of virtual machines (VMs) and configuration of services and advanced policy rules.
   - CPC Central—Provides a unified GUI that allows you to configure Policy Builder, manage custom reference table data, and start the Web-based applications and utilities.
   - Ops Center—Allows you to configure and manage the applications and pods configuration.
4. Storage Layer
   - Binding Database Client—Provisions the client to look up the PCRF Mongo Binding Database for information about the secondary key lookup across 4G and 5G.
   - MongoDB—Preserves the subscriber-specific, balance data, and admin configuration data.

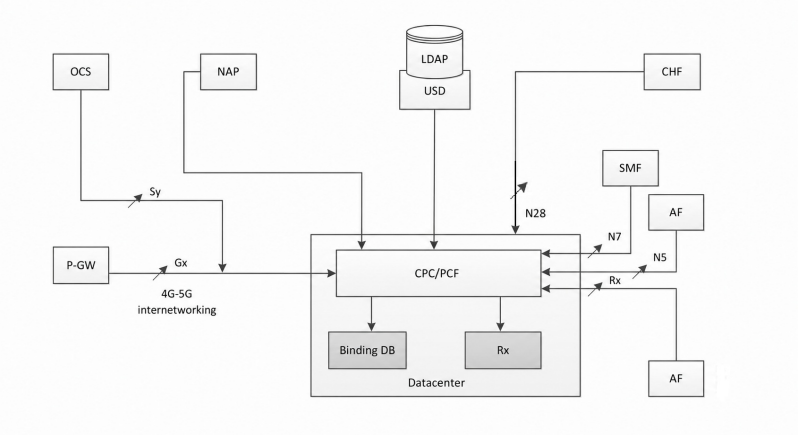
- Session store—Contains the data which CDL accesses for processing a session persistence activity. Stores the CPC sessions.
- Etcd—Contains the Diameter endpoint configurations.

CPC deployment

The CPC reduces the deployment complexity by integrating CPC and PCF in a unified environment.

This figure illustrates the CPC deployment.

Figure 2: CPC architecture components



The CPC deployment architecture includes:

- One Region = Two sites. Each site has one cluster (total two clusters in a region).
- Noncloud-native deployment along with cloud-native 5G CPC.
- External binding database is the local database for PCRF.
- MongoDB is the dedicated session database for PCRF.

The CPC's deployment architecture includes:

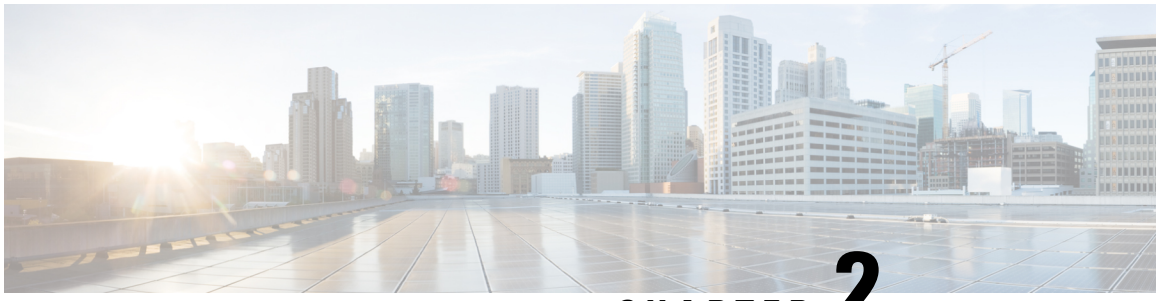
- One Region = Two sites. Each site has one cluster (total two clusters in a region).
- Cloud-native deployment that deployed along with 4G PCRF.
- Cisco CDL is the dedicated session database for CPC.

Supported Interfaces

CPC and other NFs in 5GC use these:

- Rx– Reference point for interworking with AF, PCRF, and CPC
- N5– Reference point between CPC and AF
- N7– Reference point between CPC and SMF

- N15– Reference point between CPC and AMF
- N28– Reference point between CPC and CHF
- N36– Reference point between CPC and UDR
- LDAP– Reference point between CPC and external subscriber profile



CHAPTER 2

Deploy and Configure CPC through Ops Center

Cisco CPC architecture

The CPC Ops Center allows you to configure the CPC features such as configuring the license, CPC Engine, REST endpoint, and CDL. You can also configure the NRF components that enable the interworking of various NFs.

Policy Ops Center reuses the existing Ops Center image from mobile-cnat-infrastructure, and is accessible via the ingresses that are defined by that chart.

Prerequisites

Base CPC configuration is a comprehensive setup that

- Ensure that all the virtual network functions (VNFs) are deployed.
- Run the SMI sync operation for the CPC Ops Center and Cloud Native Common Execution Environment (CN-CEE).

Deploying and Accessing CPC

This section describes how to deploy and access the CPC Ops Center.

Deploying CPC involves these steps:

1. Deploying CPC
2. Accessing the CPC Ops Center

Deploying CPC

The Subscriber Microservices Infrastructure (SMI) platform is responsible for deploying and managing the Cloud Native 5G CPC application and other network functions.

For information on how to deploy CPC Ops Center on a vCenter environment, see *Configuring the vCenter Environment* section in *Ultra Cloud Core SMI Cluster Deployer Operations Guide*.

For deploying CPC Ops Center on an OpenStack environment, see *UAME-based VNF Deployment* section in the *UAME-based 4G and 5G VNF Deployment Automation Guide, Release 6.9*.

For information on how to deploy CPC Ops Center on bare metal servers (currently Cisco UCS-C servers) environment, see *Operating the SMI Cluster Manager on Bare Metal* section in *Ultra Cloud Core Subscriber Microservices Infrastructure — Operations Guide*.

Accessing the CPC Ops Center

This section describes how to access the CPC Ops Center.

You can access the CPC Ops Center from the console application or the Web-based CLI console. Depending upon your selection, access one of the following from the master node:

1. **CLI:**

   ssh admin@*ops_center_pod_ip* **-p 2024**

2. **Web-based console:**

   a. Log in to the Kubernetes master node.

   b. To view the available ingress connections, use the following configuration:

      kubectl get ingress *namespace*

      The available ingress connections are displayed.

   c. Select the appropriate ingress from where you want to run Ops Center and open the following URL from the browser:

      cli.*namespace***-ops-center.***ip_address***.nip.io**

Sample configuration

Important The mandatory parameters are required to ensure that the critical pods such as CRD and Policy Engine are in the running state.

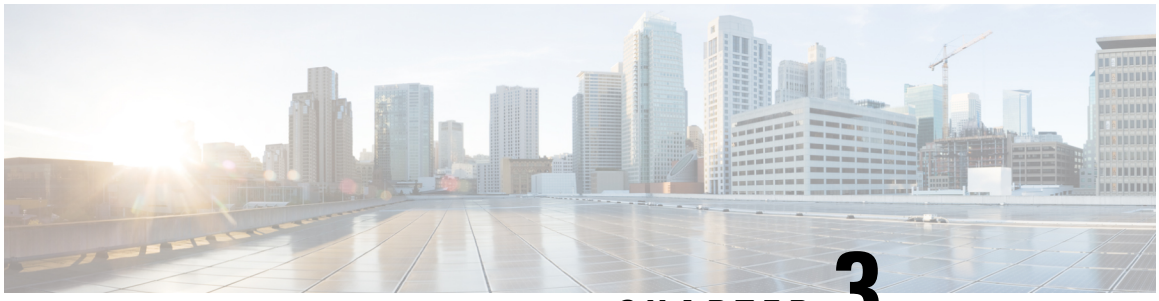
```
config
datastore primary-endpoint connection-settings keep-alive keep-alive-time-ms 200
datastore primary-endpoint connection-settings channel count 4
datastore primary-endpoint connection-settings timeout-ms 300
ldap replicas 2
ldap server-set USD
 search-user dn cn=sdcUser,dc=C-NTDB
```

```
 search-user password siemens
 health-check interval-ms 10000
 health-check dn cn=sdcUser,dc=C-NTDB
 health-check filter msisdn=918369110173
 health-check attributes napCustType
 initial-connections                              10
 max-connections                                  10
 retry-count                                      2
 retry-timer-ms                                   100
 max-failover-connection-age-ms                   60000
 binds-per-second                                 0.2
 number-consecutive-timeouts-for-bad-connection 2
 missing-attribute-result-code                    32
 connection 192.0.2.18 389
  priority        400
  connection-rule ROUND_ROBIN
  auto-reconnect  true
  timeout-ms      200
  bind-timeout-ms 3000
 exit
 connection 192.0.2.18 389
  priority        400
  connection-rule ROUND_ROBIN
  auto-reconnect  true
  timeout-ms      200
  bind-timeout-ms 3000
 exit
exit
//This is a mandatory parameter
db global-settings db-replica 3
//This is a mandatory parameter
db spr shard-count 1
db balance shard-count 1
debug tracing type DISABLED
debug splunk hec-url http://192.0.2.18:6066
debug splunk hec-token efd41999-fa22-1111-aca7-2222fb36c6af
debug splunk batch-interval-ms 100
debug splunk batch-count 10
debug splunk batch-size-bytes 102400
debug logging default-level error
diameter settings timeouts-ms dpa 5000
diameter application rx
 application-id   16777236
 tgpp-application true
 vendor           [ 10415 ]
exit
diameter group rx-pcf01-proto1
 mode server
 stack rx
  application rx
  bind-ip     192.0.2.18
  bind-port   3868
  fqdn        pcf-rx-server-1
  realm       pcf.rx.server.cisco.com
  node-host   pcf01-protocol01
 exit
exit
diameter group rx-pcf01-proto2
 mode server
 stack rx
  application rx
  bind-ip     192.0.2.18
  bind-port   3868
  fqdn        pcf-rx-server-2
```

```
  realm       pcf.rx.server.cisco.com
  node-host   pcf01-protocol02
 exit
exit
//This is a mandatory parameter
rest-endpoint ips    [ 192.0.2.18 192.0.2.19 192.0.2.20 ]
//This is a mandatory parameter
rest-endpoint port   9088
rest-endpoint tracing-service-name pcf-rest-endpoint
rest-endpoint replicas 2
rest-endpoint interface n28
 ip [ 192.0.2.18 ]
exit
rest-endpoint interface n7
 ip [ 192.0.2.19 ]
exit
rest-endpoint interface nnrf
 ip [ 192.0.2.20 ]
exit
advance-tuning overload-control rest global limits max-requests-per-sec 5000
advance-tuning overload-control rest global action throttle-action REJECT
advance-tuning overload-control diameter global limits max-requests-per-sec 5000
advance-tuning overload-control diameter global action throttle-action DROP
advance-tuning async-threading request-timeout-ms 300
api unified engine-group pcf01production
//This is a mandatory parameter
engine pcf01production
//This is a mandatory parameter
  replicas              2
//This is a mandatory parameter
  unified-api-replicas  1
//This is a mandatory parameter
  subversion-run-url    http://svn/repos/run
//This is a mandatory parameter
  subversion-config-url http://svn/repos/configuration
//This is a mandatory parameter
  tracing-service-name  pcf-engine
  grpc port   5005
  grpc externalIPs [ 192.0.2.21 ]
  properties com.cisco.engine.log.type
    value 2
 exit
 properties disableCommandClient
  value true
 exit
 properties ldap.retry.time.ms
  value 200
 exit
exit
label protocol-layer key smi.cisco.com/node-type
label protocol-layer value protocol
label service-layer key smi.cisco.com/node-type
label service-layer value service
label cdl-layer key smi.cisco.com/node-type
label cdl-layer value session
label oam-layer key smi.cisco.com/node-type
label oam-layer value oam
profile nf-client nf-type chf
 chf-profile chprofile
  locality locchf1
   priority 10000
   service name type nchf-spendinglimitcontrol
    endpoint-profile chf-profile
     capacity   10
```

```
     priority   10
     uri-scheme http
     version
      uri-version v1
      exit
     exit
     endpoint-name ep1
      primary ip-address ipv4 192.0.2.22
      primary ip-address port 5088
     exit
    exit
   exit
  exit
 exit
exit
profile nf-pair nf-type CHF
 nrf-discovery-group    discoveryGroup
 subscription-enabled   false
 subscription-extension 3
 locality client  pcf01
 locality preferred-server loc1
 locality geo-server loc2
exit
//This is a mandatory parameter
service-registration profile locality pcf01
//This is a mandatory parameter
service-registration profile plmn-list 100 010
//This is a mandatory parameter
exit
//This is a mandatory parameter
service-registration profile plmn-list 123 456
//This is a mandatory parameter
exit
//This is a mandatory parameter
service-registration profile snssais 1
//This is a mandatory parameter
 sd 123456
exit
group nf-mgmt group1
 nrf-mgmt-group registerGroup
 locality       pcf01
 failover sla 100
 reconnect interval 60000
exit
group nrf discovery discoveryGroup
 service type nrf nnrf-disc
  endpoint-profile discoveryProfile
   capacity   10
   priority   10
   uri-scheme http
   version
    uri-version v1
    exit
   exit
   endpoint-name ep1
    priority 1
    capacity 10
    primary ip-address ipv4 192.0.2.12
    primary ip-address port 8183
    secondary ip-address ipv4 192.0.2.12
    secondary ip-address port 8184
    tertiary ip-address ipv4 192.0.2.12
    tertiary ip-address port 8185
   exit
```

```
  exit
 exit
exit
group nrf mgmt registerGroup
 service type nrf nnrf-nfm
  endpoint-profile registerProfile
   capacity   20
   priority   10
   uri-scheme http
   version
    uri-version v1
    exit
   exit
   endpoint-name ep1
    priority 1
    capacity 10
    primary ip-address ipv4 192.0.2.12
    primary ip-address port 8183
    secondary ip-address ipv4 192.0.2.12
    secondary ip-address port 8184
    tertiary ip-address ipv4 192.0.2.12
    tertiary ip-address port 8185
   exit
  exit
 exit
exit
cdl system-id 1
cdl node-type session
cdl zookeeper data-storage-size 1
cdl zookeeper log-storage-size 1
cdl zookeeper replica 3
cdl logging default-log-level warn
cdl datastore session
 cluster-id 1
 endpoint replica 4
 index map 4
 slot replica 2
 slot map 10
 slot notification limit 10
exit
cdl kafka replica 3
cdl kafka storage 1
end
```



CHAPTER 3

Cisco Common Data Layer

Geographic redundancy support

The CPC extends support to the Geographic Redundancy (GR) version of the Cisco Common Data Layer (CDL). When the highest rated CDL endpoint fails, CPC attempts the same operation on the next highly rated CDL endpoint thus providing a nondisrupted message handling. If the next rated endpoint is unavailable, then CPC reattempts the operation on the subsequent endpoint that has the highest rating and so on.

For more information on the CDL concepts, see the *Ultra Cloud Core Common Data Layer Configuration Guide*.

Limitations for GR support feature

The GR support feature has these limitations:

- The CPC attempts to reroute the calls only when it encounters gRPC errors such as UNAVAILABLE. It does not acknowledge errors that the datastore returns and actual gRPC timeouts such as DEADLINE_EXCEEDED gRPC status code.
- The CPC Engine does not resolve failures occurring with the datastore such as indexing and slot failures. The CDL layer must resolve these failures and if necessary, send an API call on the remote.

How it Works

When you configure the CDL in CPC through the CPC Ops Center, CPC gets enabled to support multiple CDL datastore endpoints. You can configure the endpoints by specifying the IP addresses, port numbers, and assigning ratings to each endpoint. By default, CPC considers the local endpoint as the primary endpoint, which has the highest rating. CPC performs CDL API operations on the primary endpoint. If this endpoint is

unavailable, then CPC routes the operations to the next highest rated endpoint. CPC keeps failing over to the accessible secondary endpoint or until all the configured secondaries are exhausted. It does not reattempt a query on the next rated endpoint if the endpoint is reachable but responds with error or timeout.

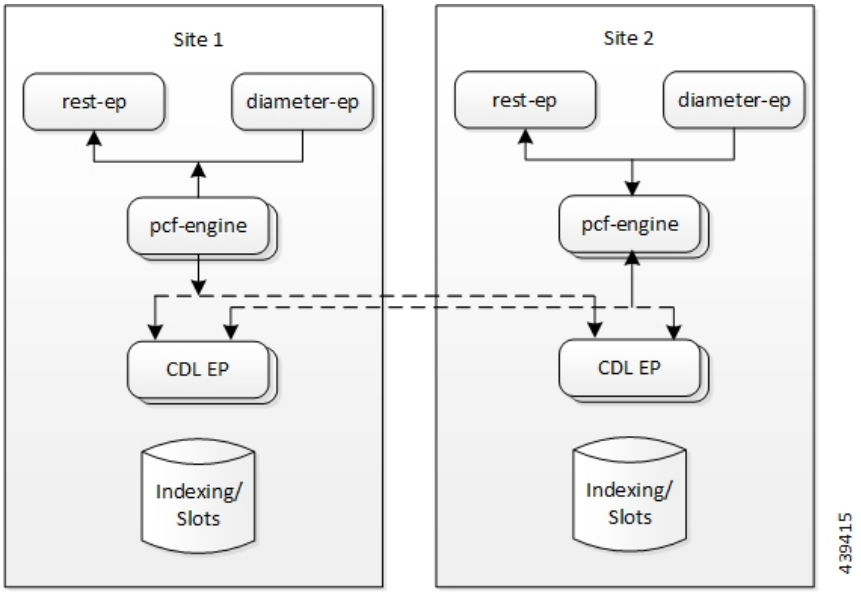
If CPC is unable to access any of the endpoints in the cluster, then CDL operation fails with the "Datastore Unavailable" error.

Architecture

You can configure CDL through CPC Ops Center. CDL in the GR mode replicates the session data across the configured sites. When CPC connects to the CDL, it always treats the local CDL endpoints as the primary endpoint and the remote endpoints as secondaries (with the appropriate rating). CPC uses the secondary endpoints when the connection to the primary endpoint fails.

This illustration depicts the failover that happens when the CPC/ PCF Engine is unable to access the primary CDL datastore endpoint.

Figure 3: CDL Datastore Architecture



Call Flows

This section describes the key call flows for this feature.

CDL endpoint failure

This section describes the CDL Endpoint Failure call flow.

Figure 4: CDL Endpoint Failure Call Flow

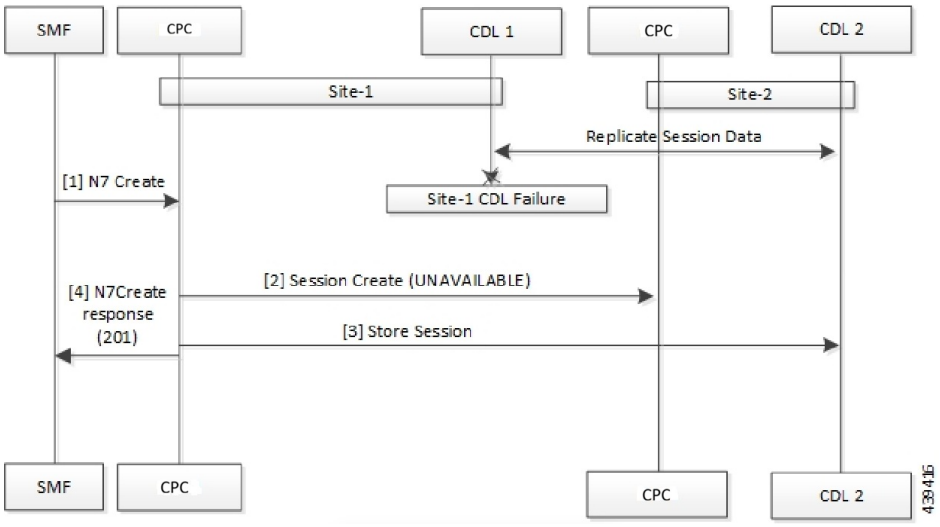


Table 1: CDL Endpoint Failure Call Flow Description

Step	Description
1	In the Site 1 environment, the SMF sends a N7 Create Request to the CPC 1 over the N7 interface.
2	The CPC 1 sends Session Create Request to the CPC 2.
3	The CPC 1 sends a Session Store Request to the CDL2.
4	The CPC 1 sends N7 Create Response to the SMF.

GR Call Flows

This section describes the possible CDL GR mode scenarios that could initiate a failover to another site.

Indexing shard failure

This section describes how the failover happens when two index replicas that belong to the same shard are down or unavailable.

The indexing shard failure is an example of two points-of-failure scenario where the two replicas reside on different virtual machines or hosts.

The CPC REST endpoint and CPC Engine redirect the traffic to the secondary CDL endpoint site (Site 2) based on the highest rating when the primary CDL site (Site 1) is unavailable.

Figure 5: Indexing Shard Failure Call Flow

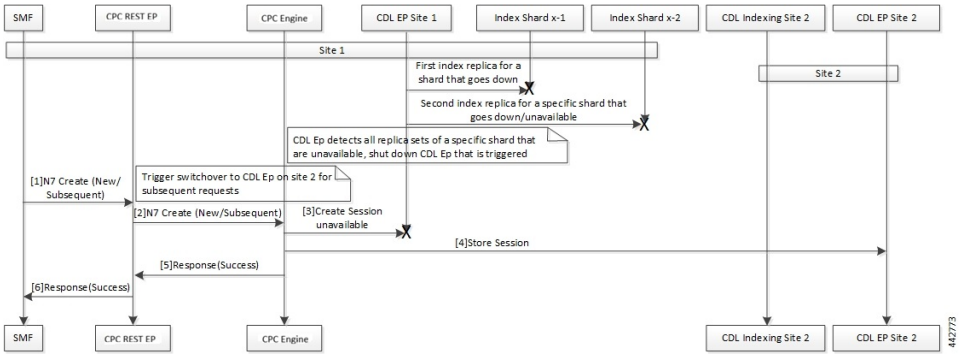


Table 2: Indexing Shard Failure Call Flow Description

Step	Description
1	In the Site 1 environment, index replica 1 and replica 2 for a configured shard has failed or unavailable. Since both the replicas for the shard are unavailable, the CDL endpoint in Site 1 is shut down and all the subsequent requests are directed to the CDL endpoint on Site 2. In the Site 1 environment, the SMF sends a Create Request to CPC REST endpoint over the N7 interface.
2	After receiving the request, the CPC REST endpoint forwards the Create Request to the CPC Engine.
3	The CPC Engine attempts to reach the CDL endpoint to send the Session Create Request. However, the CDL endpoint is unreachable. The CPC Engine sorts the CDL points across Site 1 and Site 2 to recognize the endpoint with the highest rating or priority.
4	The Create Request is evaluated in the stored session and the CPC Engine forwards the request to the CDL endpoint residing in Site 2.
5	After the call request is successful, the CPC Engine notifies the Success Message to the CPC REST endpoint.
6	The CPC REST endpoint forwards the Success Message to the SMF.

Slot replica set failure

This section describes how the failover happens when two slot replicas that belong to the same replica set are down or unavailable.

The slot failure is an example of two points-of-failure scenario where the two slot replicas reside on different virtual machines or hosts.

Figure 6: Slot Replica Set Failure Call Flow

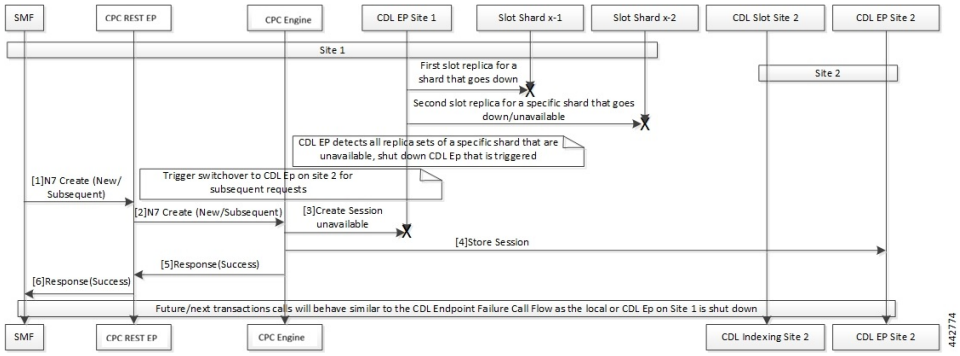


Table 3: Slot Replica Set Failure Call Flow Description

Step	Description
1	In the Site 1 environment, slot replica 1 and replica 2 for a configured shard is down or unavailable. Since both the replicas for the shard are unavailable, the CDL endpoint in Site 1 is shut down and all the subsequent requests are directed to the CDL endpoint on Site 2. In the Site 1 environment, the SMF sends a N7 Create request to CPC REST endpoint over the N7 interface.
2	The CPC REST endpoint receives the request and forwards it to the CPC Engine.
3	The CPC Engine attempts to connect the CDL endpoint to send the Session Create request. If the CDL endpoint is unreachable, the CPC Engine sorts the CDL points across Site 1 and Site 2 to recognize the endpoint with the highest rating or priority.
4	The Create Request is evaluated in the stored session and the CPC Engine forwards the request to the CDL endpoint residing in Site 2.
5	After the call request is successful, the CPC Engine notifies the Success message to the CPC REST endpoint.
6	The CPC REST endpoint forwards the Success message to the SMF.

CDL configuration through Ops Center

This section describes how to configure the CDL endpoints.

Configure the CDL using CPC Ops Center involves these steps:

- Configure the CDL Session Database and Defining the Base Configuration
- Configure Kafka in CDL
- Configure Zookeeper in CDL

Configure the CDL session database and defining the base configuration

This section describes how to configure the CDL session database and define the base configuration in CPC.

To configure the CDL session database and define the base configuration in CDL, use the following configuration in the Policy Ops Center console:

```
config
  cdl
    system-id system_id
    node-type node_type
    enable-geo-replication [ true | false ]
    zookeeper replica zookeeper_replica_id
    remote-site remote_system_id
      db-endpoint host host_name
      db-endpoint port port_number
      kafka-server remote_kafka_host1 remote_port1
      kafka-server remote_kafka_host2 remote_port2
      kafka-server remote_kafka_host3 remote_port3
      exit
   cdl logging default-log-level debug_level
      cdl datastore session
      cluster-id cluster_id
      geo-remote-site remote_site_value
      endpoint replica replica_number
      endpoint external-ip ip_address
      endpoint external-port port_number
         index map map_value
         slot replica replica_slot
         slot map map/shards
         slot write-factor write_factor
         slot notification host host_name
         slot notification port port_number
         slot notification limit tps

         index replica index_replica
         index map map/shards
         index write-factor write_factor
         end
```

NOTES:

- **system-id** *system_id*—(Optional) Specify the system or Kubernetes cluster identity. The default value is 1.
- **node-type** *node_type*—(Optional) Specify the Kubernetes node label to configure the node affinity. The default value is “session.” *node_type* must be an alphabetic string of 0-64 characters.
- **enable-geo-replication [true | false]** —(Optional) Specify the geo replication status as enable or disable. The default value is false.
- **zookeeper replica** *zookeeper_replica_id*—Specify the Zooker replica server ID.
- **remote-site** *remote_system_id*—Specify the endpoint IP address for the remote site endpoint. Configure this command only when you have set the cdl enable-geo-replication to true.

- **db-endpoint host** *host_name*—Specify the endpoint IP address for the remote site. Configure this command only when you have set the cdl enable-geo-replication to true.
- **db-endpoint port** *port_number*—Specify the endpoint port number for the remote site endpoint. The default port number is 8882. Configure this command only when you have set the cdl enable-geo-replication to true.
- **kafka-server** *remote_kafka_host1 remote_port1*—Specify the Kafka server's external IP address and port number of the remote site that the remote-system-id identifies. You can configure multiple host address and port numbers per Kafka instance at the remote site. Configure this command only when you have set the cdl enable-geo-replication to true.
- **endpoint replica** *replica_number*—(Optional) Specify the number of replicas to be created. The default value is 1. *replica_number* must be an integer in the range of 1 – 16.
- **endpoint external-ip** *ip_address*—(Optional) Specify the external IP address to expose the database endpoint. Configure this command only when you have set the cdl enable-geo-replication to true.
- **endpoint external-port** *port_number*—(Optional) Specify the external port number to expose the database endpoint. Configure this command only when you have set the cdl enable-geo-replication to true. The default value is 8882.
- **slot replica** *replica_slot*—(Optional) Specify the number of replicas to be created. The default value is 1. *replica_slot* must be an integer in the range of 1 – 16.
- **slot map** *map/shards*—(Optional) Specify the number of partitions in a slot. The default value is 1. *map/shards* must be an integer in the range of 1 – 1024.
- **slot write-factor** *write_factor*—(Optional) Specify the number of copies to be written before successful response. The default value is 1. *write_factor* must be an integer in the range of 0 – 16. Make sure that the value is lower than or equal to the number of replicas.
- **slot notification host** *host_name*—(Optional) Specify the notification server hostname or IP address. The default value is datastore-notification-ep.
- **slot notification port** *port_number*—(Optional) Specify the notification server port number. The default value is 8890.
- **slot notification limit** *tps*—(Optional) Specify the notification limit per second. The default value is 2000.
- **index replica** *index_replica*—(Optional) Specify the number of replicas to be created. The default value is 2. *index_replica* must be an integer in the range of 1 – 16.
- **index map** *map/shards*—(Optional) Specify the number of partitions in a slot. The default value is 1. *map/shards* must be an integer in the range of 1 – 1024. Avoid modifying this value after deploying the CDL.
- **index write-factor** *write_factor*—(Optional) Specify the number of copies to be written before successful response. The default value is 1. *write_factor* must be an integer in the range of 0 – 16.

Configure Kafka in CDL

This section describes how to configure Kafka in CDL.

To configure the Kafka in CDL, use these configuration:

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To configure Kafka, use these configuration:

```
config
   cdl kafka replica number_of_replicas
      enable-JMX-metrics [ true | false ]
      external-ip ip_address  port_number
      enable-persistence [ true | false ]
      storage storage_size
      retention-time retention_period
      retention-size retention_size
      end
```

NOTES:

All the following parameters are optional.

- **cdl kafka replica** *number_of_replicas*—Specify the number of replicas to be created. The default value is 3. *number_of_replicas* must be an integer in the range of 1 – 16.
- **enable-JMX-metrics [true | false]**—Specify the status of the JMX metrics. The default value is true.
- **external-ip** *ip_address port_number*—Specify the external IPs to expose to the Kafka service. Configure this command when you have set the **enable-geo-replication** parameter to true. You are required to define an external IP address and port number for each instance of the Kafka replica. For example, if the **cdl kafka replica** parameter is set to 3, then specify three external IP addresses and port numbers.
- **enable-persistence [true | false]**—Specify whether to enable or disable persistent storage for Kafka data. The default value is false.
- **storage** *storage_size*—Specify the Kafka data storage size in gigabyte. The default value is 20 GB. *storage_size* must be an integer in the range of 1-64.
- **retention-time** *retention_period*—Specify the duration (in hours) for which the data must be retained. The default value is 3. *retention_period* must be an integer in the range of 1 – 168.
- **retention-size** *retention_size*—Specify the data retention size in megabyte. The default value is 5120 MB.

Configure Zookeeper in CDL

This section describes how to configure Zookeeper in CDL.

To configure Zookeeper in CDL, use this configuration:

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To configure the parameters, use this configuration:

```
config
   cdl zookeeper data-storage-size data_storage
     log-storage-size log_storage
```

```
replica number_of_replicas
enable-JMX-metrics [ true | false ]
enable-persistence [ true | false ]
end
```

NOTES:

All these parameters are optional.

- **cdl zookeeper data-storage-size** *data_storage*—Specify the size of the Zookeeper data storage in gigabyte. The default value is 20 GB. *data_storage* must be an integer in the range of 1-64.
- **log-storage-size** *log_storage*—Specify the size of the Zookeeper data log's storage in gigabyte. The default value is 20 GB. *log_storage* must be an integer in the range of 1-64.
- **replica** *number_replicas*—Specify the number of replicas that must be created. The default value is 3. *number_replicas* must be an integer in the range of 1-16.
- **enable-JMX-metrics [true | false]**—Specify the status of the JMX metrics. The default value is true.
- **enable-persistence [true | false]**—Specify the status of the persistent storage for Zookeeper data. The default value is false.

Sample Configuration

The following is a sample configuration of CDL in the HA environment.

```
cdl system-id   system_i
cdl enable-geo-replication true
cdl zookeeper replica num_zk_replica
cdl datastore session
 endpoint replica ep_replica
index map index_shard_count
 slot replica slot_replica
 slot map slot_shard_count
exit
cdl kafka replica kafka_replica
```

Sample Configuration

These is a sample configuration of CDL in the HA environment.

```
cdl system-id   system_i
cdl enable-geo-replication true
cdl zookeeper replica num_zk_replica
cdl datastore session
 endpoint replica ep_replica
index map index_shard_count
 slot replica slot_replica
 slot map slot_shard_count
exit
cdl kafka replica kafka_replica
```

CDL endpoint configuration

CDL endpoint configuration is a process that involves configuring external services and associating the datastore with the CDL endpoint service.

Configuration steps

Configure the CDL endpoints involves these steps:

1. Configure the External Services
2. Associating the Datastore with the CDL Endpoint Service

Configure the external services

Configure the CDL endpoints that are available in the Datastore CLI node of the CPC Ops Center. By default, these endpoints are deployed but require configuration with specific IP addresses and port numbers for each CDL site and instance.

CDL gets deployed in the GR environment as part of the SMI deployment procedure. While the CDL endpoints are available by default, you must configure them with the appropriate parameters for your specific deployment.

Procedure

Step 1 Open the Policy Ops Center console and navigate to the datastore CLI.

Step 2 Configure the external service parameters using the following configuration:

Example:

```
config
    external-services site_name
    ips ip_address
    ports port_number
    end
```

NOTES:

- **external-services** *site_name*—Specify the CDL site or instance name.
- **ips** *ip_address*—Specify the IP address on which the CDL endpoint is exposed.
- **ports** *port_number*—Specify the port number on which the CDL endpoint is exposed.

The external services are configured with the specified CDL endpoints, IP addresses, and port numbers for each site and instance.

Associate the datastore with the CDL endpoint service

Configure the external service for each CDL endpoint service that you plan to use.

This configuration allows you to associate the datastore with CDL endpoint services by specifying the service name, port number, and rating priority.

Procedure

Step 1 Open the Policy Ops Center console and navigate to the datastore CLI.

Step 2 To associate the datastore with CDL endpoint service, use the following configuration:

Example:

```
config
   datastore external-endpoints service_name
   port port_number
   rating rating_priority
   end
```

NOTES:

- **datastore external-endpoints** *service_name*—Specify the service name that belongs to the external services.
- **port** *port_number*—Specify the port number where the external service resides.
- **rating** *rating_priority*—Specify the rating or priority of the external service. CPC gives preference to the endpoints with the higher ratings.

The datastore is now associated with the CDL endpoint service with the specified configuration parameters.

Feature description

The Cisco Data Layer (CDL) stale session cleanup identifies and removes duplicate sessions that share identical International Mobile Subscriber Identity (IMSI) and Access Point Name (APN) identifiers. In high-traffic environments, duplicate sessions breach session capacity limits, which causes the CDL datastore service to reject new session requests. This enhancement uses the CDL `index-overwrite-detection` mechanism to enforce a `delete-record` action, which ensures that only the most recent session record remains active.

Configure stale session cleanup

Before you begin

Ensure that you meet these requirements:

- Verify that the CDL datastore service is deployed and operational.
- Confirm that the specific key families (`FramedIpv6PrefixKey` and `SupiDnnKey`) are identified for conflict detection.

Enable index overwrite detection and configure the `delete-record` action for stale sessions.

Procedure

Step 1 Log in to the CPC Ops-Center CLI and enter `config` mode.

Step 2 Access the CDL datastore session configuration.

```
cdl datastore session
```

Step 3 Enable index overwrite detection for `FramedIpv6PrefixKey`.

```
features index-overwrite-detection unique-keys-prefix FramedIpv6PrefixKey
  action delete-record
exit
```

Step 4 Enable index overwrite detection for `SupiDnnKey`.

```
features index-overwrite-detection unique-keys-prefix SupiDnnKey
  action delete-record
exit
```

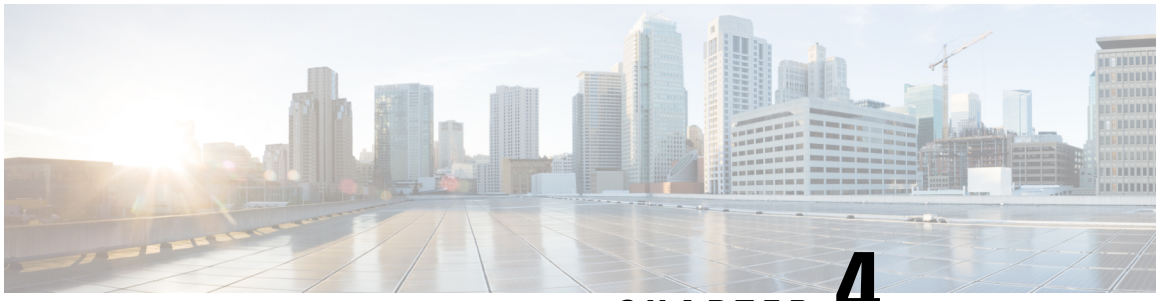
Step 5 `commit` the configuration changes.

Example:

```
cdl datastore session
 cluster-id 1
 label-config session
 features index-overwrite-detection unique-keys-prefix FramedIpv6PrefixKey
  action delete-record
 exit
 features index-overwrite-detection unique-keys-prefix SupiDnnKey
  action delete-record
 exit
exit
```

Step 6 Verify the configuration `show running-config cdl datastore session`. Confirm that the output displays `action delete-record` under the `index-overwrite-detection` configuration for both `FramedIpv6PrefixKey` and `SupiDnnKey`.

The CDL datastore service automatically monitors incoming session requests for duplicate IMSI and APN combinations. When a collision occurs, the service removes the older record and retains the latest session, which prevents the CDL from breaching session capacity limits.



CHAPTER 4

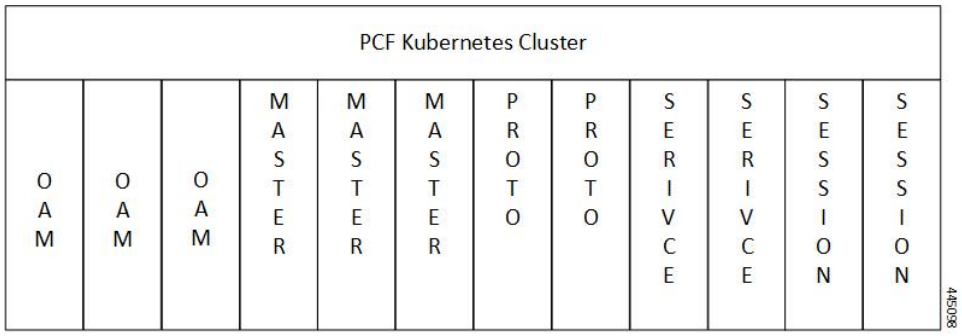
Converged Policy Software Upgrade

Software Upgrade Qualification

For high availability and fault tolerance, a minimum of two K8s worker nodes are required for each tier. You can have multiple replicas for each worker node. Kubernetes orchestrates the pods using the StatefulSets controller. The pods require a minimum of two replicas for fault tolerance.

This figure depicts a CPC K8s Cluster with 12 nodes – 3 Master nodes, 3 Operations, and Management (OAM) worker nodes, 2 Protocol worker nodes, 2 Service worker nodes, 2 Session (data store) worker nodes.

Figure 7: CPC Kubernetes Cluster



Note

- OAM worker nodes - These nodes host the Ops Center pods for configuration management and metrics pods for statistics and Key Performance Indicators (KPIs).
- Protocol worker nodes - These nodes host the CPC protocol-related pods for service-based interfaces (N5, N7, N28, N36, and NRF) and Diameter Rx Endpoint.
- Service worker nodes - These nodes host the CPC application-related pods that perform session management processing.
- Session worker nodes - These nodes host the database-related pods that store subscriber session data.

Update CPC

The CPC software update or in-service update procedure utilizes the K8s rolling strategy to update the pod images. In K8s rolling update strategy, the pods of a StatefulSet are updates sequentially to ensure that the ongoing process remains unaffected. Initially, a rolling update on a StatefulSet causes a single pod instance to terminate. A pod with an updated image replaces the terminated pod. This process continues until all the replicas of the StatefulSet are updated. The terminating pods exit gracefully after competing all the ongoing processes. Other in-service pods continue to receive and process the traffic to provide a seamless software update. You can control the software update process through the Ops Center CLI.

Note Each pod needs a minimum of two replicas for high availability. For example, Policy Engine must have 2 Engine replicas. In a worst-case scenario, the processing capacity of the pod may briefly reduce to 50% while the software update is in-progress.

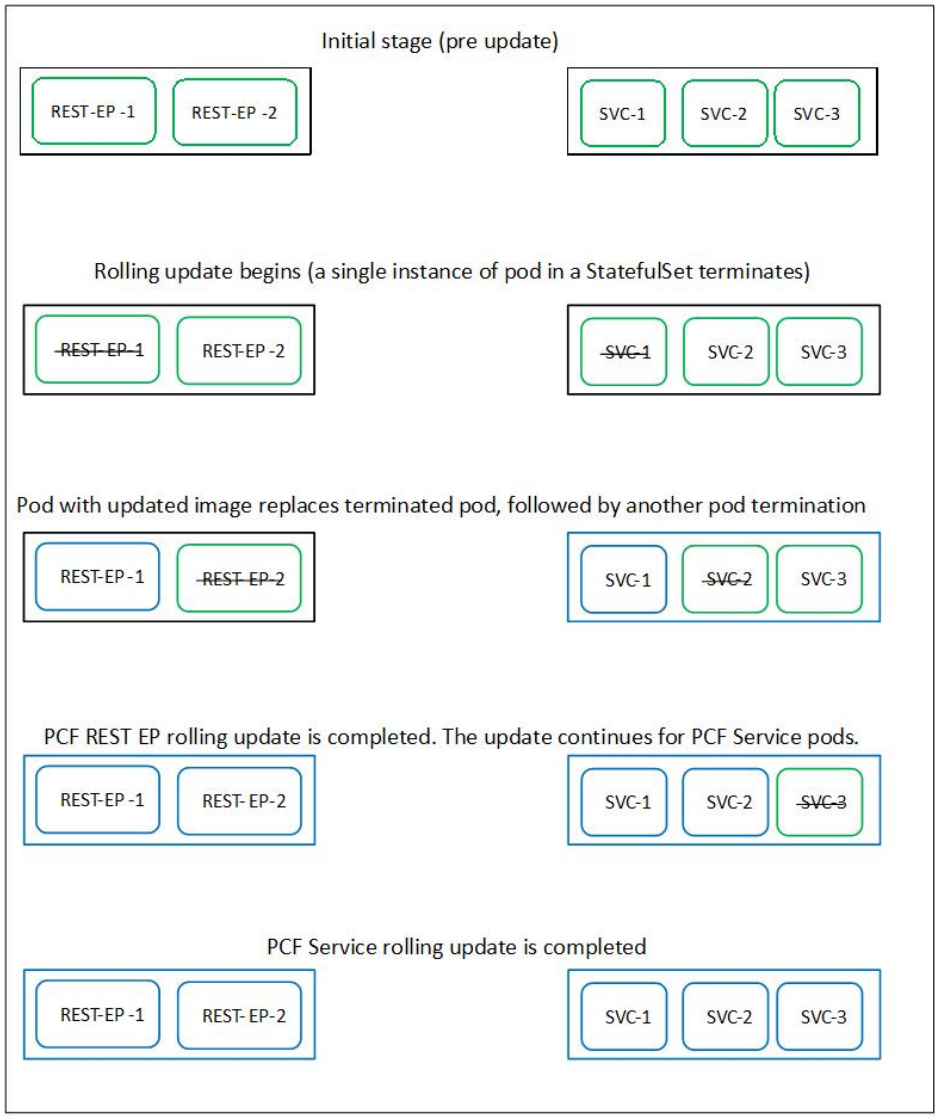
The following figure illustrates a CPC rolling update for CPC REST endpoint pods (two replicas) on Protocol worker nodes along with CPC Service pods (three replicas) on Service worker nodes.

Summary

These are the three stages of deploying and validating CPC converged policy software:

Workflow

Figure 8: CPC Rolling Update



1. Stage 1 — Pod termination: the rolling update on a StatefulSet causes a single pod instance to terminate. The terminating pod exits gracefully after completing all ongoing processes.
2. Stage 2 — Sequential update: after the first pod is updated and healthy, the next pod in the StatefulSet terminates and updates. This process continues until all replicas of the StatefulSet are updated.
3. Stage 3 — Completion: once all replicas across all tiers are updated, the cluster sync is complete and the new software version is active. The figure illustrates a converged policy rolling update for Diameter endpoint pods (two replicas) on Protocol worker nodes and CPC converged policy Service pods (three replicas) on Service worker nodes.

Upgrade prerequisites

System readiness requirements

Ensure that all nodes, including all the pods in each node, are up and running before starting the upgrade. Also ensure that a patch version of the CPC Converged Policy software is available.

- Only patch upgrades support the rolling update strategy. Major version upgrades do not support rolling upgrades.
- CPC Converged Policy uses the versioning format YYYY.RN.MN, where YYYY is the release year, RN is the release number, and MN is the maintenance number.

Applies to all rolling software upgrades.

Starting the upgrade with nodes or pods in a non-running state increases the risk of traffic disruption and upgrade failure. Major version upgrades require a different upgrade procedure that is not covered by the rolling update strategy.

The upgrade proceeds without disruption to ongoing traffic, and all replicas update sequentially as expected.

CPU usage threshold

Trigger the rolling upgrade only when the CPU usage of all nodes is less than 50%.

- High CPU usage during an upgrade increases the risk of pod restart failures and prolonged upgrade duration.
- Monitor node CPU usage before initiating the upgrade and defer the upgrade if usage exceeds 50%.

Applies at the time the cluster sync command is issued to trigger the rolling upgrade.

The rolling update terminates and restarts pods sequentially. High CPU usage at that moment can prevent new pods from becoming healthy in time, stalling the upgrade.

The rolling upgrade completes within the expected time window without pod restart failures.

If CPU usage exceeds 50%, defer the upgrade until usage returns to an acceptable level. Investigate and resolve the cause of high CPU usage before proceeding.

Perform a health check

Use this procedure to verify the health of the CPC Converged Policy cluster before starting the upgrade.

Procedure

Step 1 Log in to the master node.

Step 2 Run the following commands to view pod and node configurations.

View pods in the SMI namespace:

Example:

```
kubectl get pods -n smi
```

View Kubernetes nodes:

Example:

```
kubectl get nodes
```

View pods in all namespaces with wide output:

Example:

```
kubectl get pod --all-namespaces -o wide
```

View pods in the CPC namespace with wide output:

Example:

```
kubectl get pods -n <cpc namespace> -o wide
```

View pods in the CEE namespace with wide output:

Example:

```
kubectl get pods -n <cee namespace> -o wide
```

List Helm releases:

Example:

```
helm list
```

Count all pods across all namespaces:

Example:

```
kubectl get pods -A | wc -l
```

Note

- If any pod is not in a running state, verify the status of the nodes using this command:

  ```
  kubectl get nodes --show-labels
  ```

- Start the CPC Ops Center with 100% functionality before starting traffic on it.

Step 3 Check the CPC Ops Center status.

Use this command to view the system status:

Example:

```
[m13-cpc/m13] cpc# show system

Tue Apr  15 13:13:03.130 UTC+00:00
system uuid 6301d096-a1d7-442d-8b6b-9577143dd47f
system status deployed true
system status percent-ready 100.0
system ops-center repository https://charts.<Master_VIP>.nip.io/cpc.2025.02.0.i64
system ops-center-debug status true
system synch running true
system synch pending false
```

Step 4 Verify that no pods are in a non-running state.

Use this command to list all pods and filter for those not in a running status:

Example:

```
kubectl get pods -A | grep -iv running
```

Sample output showing a pod in a non-running state:

```
NAMESPACE    NAME                                                              READY   STATUS
       RESTARTS         AGE
```

```
<namespace>  crd-api-<namespace>-engine-app-production-rjio-7689c94786-kvd26   1/2     CrashLoopBackOff
   17 (4m28s ago)   76m
```

If the output lists any pods, investigate and resolve the issue before proceeding with the upgrade.

All nodes are in the ready state and all services are running. The cluster is healthy and ready for the upgrade.

What to do next

Ensure that all nodes are in the ready state before proceeding. Use **kubectl get nodes** to display node states.

Prepare for upgrade

Use this procedure to back up the CPC Converged Policy configuration and deployment files and to prepare the deployment directory before initiating the upgrade.

Before you begin

Ensure that all nodes and pods are running and healthy. For more information, refer to the *Perform a health check* section.

Procedure

Step 1 Log in to the SMI Cluster Manager node as an **ubuntu** user.

Step 2 Create a new directory for the deployment files.

Example:

```
test@smicpc-cm01:~$ mkdir -p "temp_$(date +'%m%d%Y_T%H%M')" && cd "$_"
```

Step 3 Move all the CPC Converged Policy deployment files into the newly created deployment directory.

Step 4 Untar the CPC Converged Policy deployment file.

Example:

```
test@smi1cpc01-cm01:~/temp_08072019_T1651$ tar -xzvf cpc.2025.01.SPA.tgZ
./
./cpc_REL_KEY-CCO_RELEASE.cer
./cisco_x509_verify_release.py
./cpc.2020.01.0-1.tar
./cpc.2020.01.0-1.tar.signature.SPA
./cpc.2020.01.0-1.tar.SPA.README
```

Step 5 Verify the downloaded image.

Example:

```
test@smi1CPC Converged Policy01-cm01:~/temp_08072019_T1651$ cat cpc.2025.01.SPA.tgZ
```

Important
Follow the procedure in the SPA.README file to verify the build before proceeding to the next step.

These are the mandatory options for backup configuration:

Table 4: Mandatory options for backup configuration

Option	Description
backup-type	Specifies the type of backup to perform (all, PB, CRD, Ops-center).
password	Password for authentication of PB or CRD.
scp-server-dest-backup-path	Destination path for backup files on the SCP server.
scp-server-user-ip	Host IP address of the SCP server.
scp-server-user-name	Username for the SCP server.
scp-server-user-password	Password for the SCP server user.
svn-url	SVN URL for Policy Builder export — for example, **http://svn/repos//configuration**.
username	Username for authentication of PB or CRD.
schedule-backup-daily	Specifies the hour of the day when the backup runs (values range from 00 to 23).
schedule-backup-weekly	Specifies the day of the week when the backup runs (values range from 1 to 7, starting with 1 for Monday).

The deployment directory is ready and the downloaded image is verified. You can now proceed with staging the new CPC Converged Policy image.

Auto-backup of PB, CRD, and Ops Center configurations

The auto-backup utility is an Ops Center command-line tool that consolidates the backup of critical configurations within the CPC system into

- a single-command backup for Ops Center, Policy Builder, and CRD configurations — replacing the separate API commands previously required for each,
- remote storage support through Secure Copy Protocol (SCP) for off-system backup, and
- scheduled automated backups that reduce the risk of data loss and eliminate manual backup tasks.

The utility was introduced in release 2025.03.0. It improves backup management, helps prevent data loss, and enables restoring critical configuration when needed.

Backup types supported by the auto-backup utility

Configure the **backup-type** parameter to specify what the utility backs up:

- **all**: backs up PB, CRD, and Ops Center configurations in a single operation.
- **PB**: backs up only the Policy Builder configuration.

- **CRD**: backs up only the Custom Reference Data.
- **Ops-center**: backs up only the Ops Center configuration.

Back up CEE and CPC Ops Center configuration

Create manual backups of the CEE Ops Center and CPC Ops Center configuration files so you can restore them if the upgrade fails.

Procedure

Step 1 Log in to the master node as an **ubuntu** user.

Step 2 Create a directory to store the backup files.

Example:

```
ubuntu@master01:~$ mkdir backups_$(date +'%m%d%Y_T%H%M') && cd "$_"
```

Step 3 Back up the CPC Ops Center configuration and verify the line count.

Example:

```
ubuntu@master01:~/backups_08072019_T1651$ ssh -p 2024 admin@$(kubectl get svc -n $(kubectl get
namespaces | \
grep -oP 'cpc-[^ ]+') | grep ops-center | awk '{print $3}') \
"show running-config" | tee cpcops.txt | wc -l
```

Sample output:

```
3422
```

Step 4 Back up the CEE Ops Center configuration and verify the line count.

Example:

```
ubuntu@master01:~/backups_08072019_T1651$ ssh -p 2024 admin@$(kubectl get svc -n $(kubectl get
namespaces | \
grep cee) | grep ops-center | awk '{print $3}') \
"show running-config" | tee ceeops.txt | wc -l
```

Sample output:

```
874
```

Step 5 Move the SMI Ops Center backup file into the backups directory.

Example:

```
ubuntu@master01:~/backups_08072019_T1651$ scp ubuntu@$(kubectl get nodes \
-o jsonpath='{.items[0].status.addresses[?(@.type=="InternalIP")].address}'):/data/smi-backups/* .
```

Step 6 Verify the line count of the backup files.

Example:

```
ubuntu@master01:~/backups_08072019_T1651$ wc -l *
```

Sample output:

```
 874 ceeops.txt
3422 cpcops.txt
   1 popcf-cfg-backup_110219-053530.tar.gz
```

The CEE Ops Center and CPC Ops Center configuration files are backed up in the backups directory on the master node. These files can be used to restore configuration if the upgrade fails.

Back up SVN, Policy, and CRD data

Back up SVN, Policy Builder, and CRD data to preserve critical configuration before the upgrade.

Procedure

Step 1 Log in to the master node as an **ubuntu** user.

Step 2 Retrieve the Policy Builder URL.

Example:

```
kubectl get ing -n $( kubectl get namespaces | grep -oP 'cpc-(\d+|\w+)' | cut -d\  -f1) | grep
policy-builder | awk '{ print $2 }'
```

Sample output:

```
pb.cpc-02-cpc-engine-app-blv02.<ipv4address>.nip.io
```

Step 3 Navigate to the Policy Builder home page using the URL from the previous step.

Step 4 Click **Import/Export**.

Step 5 Click **All Data** and provide the following details:

- **Export URL**: specify the export URL.
- **Export File Prefix**: specify an appropriate name for the export file.

Step 6 Click **Export**.

Important
The exported file is saved to your local **Downloads** directory.

Step 7 Navigate to the Policy Builder home page to back up the CRD data.

Step 8 Click **Custom Reference Data**.

Step 9 Click **Import/Export CRD data** and then click **Export**.

Important
The CRD data is saved to your web browser's **Downloads** directory.

SVN, Policy Builder, and CRD data are backed up and available for restoration if the upgrade fails.

Configure auto-backup

Use this procedure to configure the auto-backup utility parameters from the Ops Center CLI.

Procedure

Step 1 Enter configuration mode and configure the backup parameters under **debug backup-config**.

Use the following mandatory options to configure the backup:

Table 5: Mandatory options for debug backup-config

Option	Description
backup-type	Specifies the type of backup to perform (all, PB, CRD, Ops-center).
password	Password for authentication of PB or CRD.
scp-server-dest-backup-path	Destination path for backup files on the SCP server.
scp-server-user-ip	Host IP address of the SCP server.
scp-server-user-name	Username for the SCP server.
scp-server-user-password	Password for the SCP server user.
svn-url	SVN URL for Policy Builder export — for example, **http://svn/repos//configuration**.
username	Username for authentication of PB or CRD.
schedule-backup-daily	Specifies the hour of the day when the backup runs (values range from 00 to 23).
schedule-backup-weekly	Specifies the day of the week when the backup runs (values range from 1 to 7, starting with 1 for Monday).

Sample configuration command:

Example:

```
[cpc_ops_center/cpc] cpc# config
[cpc_ops_center/cpc] cpc(config)# debug backup-config
[cpc_ops_center/cpc] cpc(config-backup-config)# backup-type all
[cpc_ops_center/cpc] cpc(config-backup-config)# username admin
[cpc_ops_center/cpc] cpc(config-backup-config)# password <pb_crd_password>
[cpc_ops_center/cpc] cpc(config-backup-config)# svn-url http://svn/repos/configuration
[cpc_ops_center/cpc] cpc(config-backup-config)# scp-server-user-ip <scp_ip_address>
[cpc_ops_center/cpc] cpc(config-backup-config)# scp-server-user-name <scp_username>
[cpc_ops_center/cpc] cpc(config-backup-config)# scp-server-user-password <scp_password>
[cpc_ops_center/cpc] cpc(config-backup-config)# scp-server-dest-backup-path /home/<scp_username>/backup
[cpc_ops_center/cpc] cpc(config-backup-config)# schedule-backup-daily 00
[cpc_ops_center/cpc] cpc(config-backup-config)# commit
[cpc_ops_center/cpc] cpc(config-backup-config)# end
```

Step 2 Verify the configuration using **show running-config debug backup-config**.

Example:

```
[cpc_ops_center/cpc] cpc# show running-config debug backup-config
```

Sample output:

```
debug backup-config
 backup-type    all
 username       admin
 password       $8$<encrypted_password>
 svn-url        http://svn/repos/configuration
 scp-server-user-ip        <scp_ip_address>
 scp-server-user-name      <scp_username>
 scp-server-user-password  $8$<encrypted_password>
 scp-server-dest-backup-path /home/<scp_username>/backup
 schedule-backup-daily  00
!
```

The auto-backup utility is configured. Backups run automatically at the scheduled time and can also be triggered manually.

Trigger an on-demand backup

To trigger a manual backup immediately from the Ops Center CLI, run this command:

```
[cpc_ops_center/cpc] cpc# debug backup now
```

You can also trigger a backup using the REST API. For more information, see the cpc Ops Center API reference.

Deploy CPC Converged Policy

Stage a new CPC image

Use this procedure to download, verify, and stage the new CPC software image on the SMI Cluster Manager node.

Procedure

Step 1 Download and verify the new CPC software image from the Cisco software download site.

Verify the SHA-256 hash of the downloaded image to confirm it has not been corrupted:

Example:

```
sha256sum cpc.<new_version>.SPA.tgZ
```

Step 2 Log in to the SMI Cluster Manager node as an **ubuntu** user.

Step 3 Copy the new CPC image to the SMI Uploads directory.

Example:

```
sudo mv cpc.<new_version>.SPA.tgZ /data/software/uploads
```

Note
The SMI Cluster Manager polls the `/data/software/uploads` directory and automatically processes new image files.

Step 4 Wait 30 seconds and verify that the image has been picked up from the Uploads directory.

Example:

```
sleep 30; ls /data/software/uploads
```

The Uploads directory should be empty after the SMI processes the image. If the file still appears, wait an additional 30 seconds and check again.

Step 5 Verify that the image has been processed and is available in the packages directory.

Example:

```
sudo du -sh /data/software/packages/*
```

Sample output:

```
1.5G  /data/software/packages/cpc.<previous_version>
1.6G  /data/software/packages/cpc.<new_version>
```

Confirm that the new version directory is present. The SMI Cluster Manager unpacks the image tarball into the packages directory.

The new CPC software image is staged and available in the SMI packages directory. You can now trigger the rolling software upgrade.

Trigger the rolling software upgrade

Trigger the rolling upgrade through the SMI Cluster Manager Ops Center to update all pod replicas sequentially to the new software version.

Before you begin

- The new CPC image is staged in the SMI packages directory. For more information, refer to the *Stage a new CPC image* section
- CPU usage on all nodes is below 50%.

Procedure

Step 1 Log in to the SMI Cluster Manager console.

Step 2 Log in to the SMI Ops Center.

Example:

```
ssh -p 2024 admin@<SMI_Cluster_Manager_IP>
```

Sample output:

```
SMI Cluster Manager
Copyright 2019-2023 Cisco Systems, Inc.
```

```
All rights reserved.
[SMI Cluster Manager] smi#
```

Step 3 Download the latest software package using the **software-packages download** command.

Specify the URL of the new CPC software tarball:

Example:

```
[SMI Cluster Manager] smi# software-packages download url <URL_of_new_cpc_image>
```

Note
The **software-packages download url** command downloads software packages from the specified HTTP or HTTPS location to the SMI Cluster Manager node.

Step 4 Verify that the tarball packages are loaded.

Example:

```
[SMI Cluster Manager] smi# software-packages list
```

Note
The **software-packages list** command lists all available software packages with their versions and current status.

Sample output:

```
NAME                    VERSION         STATUS
cpc                     2025.01.0       Active
cpc                     2025.02.0       Inactive
```

Step 5 Update the product repository URL with the latest CPC Converged Policy version in the cluster configuration.

Replace *cluster_name* and *app_name* with the values for your environment, and replace *new_version* with the new software version string:

Example:

```
[SMI Cluster Manager] smi# config
[SMI Cluster Manager] smi(config)# clusters <cluster_name> ops-centers <app_name> <instance_name>
repository <new_repository_url>
[SMI Cluster Manager] smi(config-ops-centers-<app_name>/<instance_name>)# commit
[SMI Cluster Manager] smi(config-ops-centers-<app_name>/<instance_name>)# end
```

- **clusters** *cluster_name*: name of the cluster whose nodes and configurations you want to deploy.
- **ops-centers** *app_name instance_name*: the Ops Center application name and instance name for which the repository is updated.
- **repository** *url*: the Helm chart repository URL pointing to the new CPC Converged Policy version.

Step 6 Run the cluster sync command to trigger the rolling upgrade.

Example:

```
[SMI Cluster Manager] smi# clusters <cluster_name> actions sync run
```

Important
The cluster sync command triggers the rolling upgrade immediately. Ensure all prerequisites are met and all configuration changes are committed before running this command.

- **clusters** *cluster_name*: specifies the cluster that contains the nodes to deploy.
- **actions**: configures the actions to perform on the specified cluster.

- **sync run**: triggers the cluster synchronization and initiates the rolling upgrade.

The SMI Cluster Manager begins the rolling upgrade. Monitor the upgrade progress. For more information, refer to the *Monitor the upgrade* section.

Note
If the node type is not identified during the sync, SMI may display a warning about deprecated node-type parameter usage. This warning does not affect the upgrade and can be disregarded.

The cluster sync is initiated and the rolling upgrade is underway. The SMI Cluster Manager updates pod replicas sequentially across all tiers.

Monitor the upgrade

Monitor the rolling upgrade from the SMI Cluster Manager Ops Center to confirm that the cluster sync is progressing and to detect any issues.

Procedure

Step 1 Log in to the SMI Cluster Manager Ops Center and run the cluster sync command with debug output enabled.

Example:

```
[SMI Cluster Manager] smi# clusters <cluster_name> actions sync run upgrade-strategy concurrent debug
 true
```

- **clusters** *cluster_name*: specifies the cluster whose nodes are being deployed.
- **actions**: configures the actions to be run on the cluster.
- **sync run**: triggers the cluster synchronization.
- **debug true**: enables debug mode to display detailed sync log output.

Step 2 View the current sync logs to monitor the upgrade in real time.

Example:

```
[SMI Cluster Manager] smi# clusters <cluster_name> actions sync logs
```

Note
The **sync logs** parameter displays the current synchronization logs for the specified cluster.

Step 3 Monitor the sync logs continuously using the **monitor sync-logs** command.

Example:

```
[SMI Cluster Manager] smi# monitor sync-logs <cluster_name>
```

Sample output showing a successful cluster sync:

```
2025-02-14 09:43:22,973 p=9829 u=ubuntu n=ansible | PLAY RECAP
2025-02-14 09:43:22,973 p=9829 u=ubuntu n=ansible | master01  : ok=243  changed=2    unreachable=0
   failed=0    skipped=115  rescued=0    ignored=1
```

```
2025-02-14 09:43:22,973 p=9829 u=ubuntu n=ansible | master02  : ok=190  changed=2    unreachable=0
   failed=0    skipped=139  rescued=0    ignored=0
2025-02-14 09:43:22,973 p=9829 u=ubuntu n=ansible | master03  : ok=191  changed=2    unreachable=0
   failed=0    skipped=139  rescued=0    ignored=0
Cluster sync successful
```

Note
The **monitor sync-logs** parameter continuously streams the cluster sync process for the specified cluster. Press **Ctrl+C** to stop monitoring.

Step 4 Check the overall sync status after the upgrade completes.

Example:

```
[SMI Cluster Manager] smi# clusters <cluster_name> actions sync status
```

Note
The **sync status** parameter displays the current synchronization status for the specified cluster.

Step 5 Exit the Ops Center.

Example:

```
[SMI Cluster Manager] smi# end
```

The cluster sync logs confirm that the rolling upgrade has completed successfully across all nodes.

What to do next

View detailed pod information after the upgrade through the CEE Ops Center UI. Verify the Helm status and pod health after the upgrade. See *Verify the Helm status*.

Verify the Helm status

Confirm that all Helm charts are running at the new version and that none are in a failed or pending state.

Procedure

Step 1 Run the following command on the master node to view the list of all deployed Helm charts.

Example:

```
helm list -A
```

Sample output (filtered for cpc namespace):

```
NAME                              NAMESPACE                       REVISION  UPDATED
                STATUS    CHART                                   APP VERSION
cpc-engine-app                cpc-02-cpc-engine-app-blv02   5         2025-02-14 12:04:17.898 +0000
 UTC     deployed  cpc-engine-app-2025.02.0.BUILD_20250214092901   2025.02.0.BUILD_20250214092901
cpc-network-policy            cpc-02-cpc-engine-app-blv02   2         2025-02-14 11:48:23.422 +0000
 UTC     deployed  cpc-network-policy-1.0.0                        1.0.0
crd-api-cpc-engine-app        cpc-02-cpc-engine-app-blv02   5         2025-02-14 12:04:17.896 +0000
 UTC     deployed  crd-api-2025.02.0.BUILD_20250214092901            2025.02.0.BUILD_20250214092901
policy-builder-cpc-engine-app  cpc-02-cpc-engine-app-blv02   5         2025-02-14 12:04:17.897 +0000
 UTC     deployed  policy-builder-2025.02.0.BUILD_20250214092901    2025.02.0.BUILD_20250214092901
```

```
svn-cpc-engine-app            cpc-02-cpc-engine-app-blv02   5         2025-02-14 12:04:17.897 +0000
 UTC     deployed  svn-2025.02.0.BUILD_20250214092901             2025.02.0.BUILD_20250214092901
```

Confirm that all relevant Helm releases show a status of **deployed** and that the chart version and app version match the new software version.

Step 2 If the Helm chart is not found in the output, log in to the Ops Center and run the **show helm charts** command in operational mode.

Example:

```
show helm charts
```

Sample output for version:

```
NAME                            VERSION
cpc-engine-app                  2025.02.0.BUILD_20250214092901
cpc-network-policy              1.0.0
crd-api-cpc-engine-app          2025.02.0.BUILD_20250214092901
policy-builder-cpc-engine-app   2025.02.0.BUILD_20250214092901
svn-cpc-engine-app              2025.02.0.BUILD_20250214092901
```

Sample output for status:

```
NAME                            STATUS
cpc-engine-app                  deployed
cpc-network-policy              deployed
crd-api-cpc-engine-app          deployed
policy-builder-cpc-engine-app   deployed
svn-cpc-engine-app              deployed
```

Confirm that all Helm charts show a status of **deployed** and that the version matches the upgraded software version.

All Helm charts are deployed at the new software version. The upgrade is confirmed at the Helm level.

Verify the pods

Verify the status of all CPC Converged Policy pods to confirm that the upgrade completed without leaving any pods in a non-running state.

Procedure

Step 1 View detailed information about a specific pod to investigate its status.

Example:

```
kubectl describe pod <pod_name> -n <namespace>
```

Step 2 List all pods that are not in a running state across all namespaces.

Example:

```
kubectl get pods -A | grep -v Running
```

If the output is empty, all pods are running. If any pods are listed, investigate them using **kubectl describe pod**.

Note

A pod is healthy when the **Status** column shows **Running** and the **Ready** column shows the same number on both sides of the forward slash — for example, **3/3**.

All CPC Converged Policy pods are confirmed to be in a running and ready state. The upgrade is complete.

Rolling back the upgrade

Rolling back the upgrade is the process of reversing a CPC Converged Policy upgrade by restoring the previously backed-up Ops Center, CEE, Policy Builder, and CRD configurations when

- one or more pods fail to reach a running state after the upgrade,
- the Helm chart status shows errors after the cluster sync completes, or
- critical services are unavailable and cannot be restored through standard troubleshooting.

The rollback procedure consists of three tasks performed in sequence: reload the Ops Center configuration from the backup file, update the Ops Center with the corrected configuration, and restore the Policy Builder and CRD data.

Reload Ops Center configuration

Restore the Ops Center configuration from a backup file as the first step in rolling back the upgrade.

Before you begin

Ensure that the backup files created before the upgrade are available on the master node in the `~/backups_<timestamp>` directory.

Procedure

Step 1 Log in to the master node as an **`ubuntu`** user.

Step 2 Extract the SMI backup tarball and navigate to the extracted directory.

Example:

```
cd ~/backups_<timestamp> && tar -zxf popcf-cfg-backup_<timestamp>.tar.gz
```

Step 3 Navigate to the extracted backup directory.

Example:

```
cd popcf-cfg-backup_<timestamp>
```

Step 4 Convert the exported Ops Center configuration into a clean file ready for import.

This command strips the leading path from the configuration entries so the file can be pasted directly into the Ops Center CLI:

Example:

```
perl -ne 'next if /^!|^$/; s/^\s*//; s/\s*$//; print "$_\n"' cpcops.txt > cpcops_clean.txt
```

The file `cpcops_clean.txt` is created in the current directory and is ready to be pasted into the Ops Center CLI in the next task.

The Ops Center backup is extracted and converted to a clean configuration file. Proceed to update the Ops Center configuration.

Update the Ops Center configuration

Apply the restored configuration file to the Ops Center and update the Helm repository URL to point to the previous software version.

Before you begin

Complete the Reload Ops Center configuration task. The `cpcops_clean.txt` file must be available on the master node.

Procedure

Step 1 Log in to the master node as an **`ubuntu`** user.

Step 2 Log in to the Ops Center CLI.

Example:

```
ssh -p 2024 admin@$(kubectl get svc -n $(kubectl get namespaces | grep -oP 'cpc-[^ ]+') | grep
ops-center | awk '{print $3}')
```

Step 3 Enter configuration mode and paste the contents of `cpcops_clean.txt` into the CLI session.

Important
Review the pasted configuration and verify that all sections are correctly imported. If any sections are not properly imported, manually re-enter the affected configuration lines and commit again.

Example:

```
[cpc_ops_center/cpc] cpc# config
[cpc_ops_center/cpc] cpc(config)# <paste contents of cpcops_clean.txt here>
[cpc_ops_center/cpc] cpc(config)# commit
[cpc_ops_center/cpc] cpc(config)# end
```

Step 4 Update the Helm repository URL to point to the previous CPC Converged Polcy software version.

Replace *cluster_name*, *app_name*, *instance_name*, and *previous_version* with the values from your environment:

Example:

```
[SMI Cluster Manager] smi# config
[SMI Cluster Manager] smi(config)# clusters <cluster_name> ops-centers <app_name> <instance_name>
repository https://charts.<Master_VIP>.nip.io/cpc.<previous_version>
[SMI Cluster Manager] smi(config-ops-centers-<app_name>/<instance_name>)# commit
[SMI Cluster Manager] smi(config-ops-centers-<app_name>/<instance_name>)# end
```

The Helm repository URL is updated to reference the previous software version. Run a cluster sync to apply the rollback. For more information, refer to the *Trigger the rolling software upgrade* section for the cluster sync command.

The Ops Center is updated with the restored configuration and the Helm repository URL points to the previous software version. Proceed to restore the Policy Builder and CRD data.

Restore the configuration from backup

Restore the Policy Builder policy data and CRD data from backup to return the CPC Converged Policy system to its pre-upgrade configuration state.

Before you begin

The Policy Builder and CRD backup files must be available in your local **Downloads** directory from the backup task performed before the upgrade.

Procedure

Step 1 Log in to the master node as an **ubuntu** user.

Step 2 Retrieve the Cisco Policy Suite Central URL.

Example:

```
kubectl get ing -n $(kubectl get namespaces | grep -oP 'cpc-[^ ]+') | grep cps-central | awk '{print
 $3}'
```

Sample output:

```
cps.cpc-02-cpc-engine-app-blv02.<ipv4address>.nip.io
```

Step 3 Navigate to the Cisco Policy Suite Central URL in a web browser and log in.

Step 4 Click **Import/Export** and select the **Import** tab.

Step 5 Select the exported policy data file from the backup.

Step 6 In the **Import URL** field, specify the SVN repository URL:

Example:

```
http://svn/repos/configuration
```

Step 7 Enter a description in the **Commit Message** field and click **Import**.

Step 8 Log in to the master node as an **ubuntu** user.

Step 9 Retrieve the Policy Builder URL.

Example:

```
kubectl get ing -n $(kubectl get namespaces | grep -oP 'cpc-[^ ]+') | grep policy-builder | awk
'{print $3}'
```

Sample output:

```
pb.cpc-02-cpc-engine-app-blv02.<ipv4address>.nip.io
```

Step 10 Navigate to the Policy Builder URL and click **Build Policies using version controlled data**.

Step 11 Select the repository from the drop-down list and click **OK**.

Step 12 Click **Publish to Runtime Environment**, enter a description in the **Commit Message** field, and click **OK**.

The Policy Builder configuration is restored and published to the runtime environment.

Step 13 Restore the CRD data: in the Cisco Policy Suite Central home page, click **Custom Reference Data**.

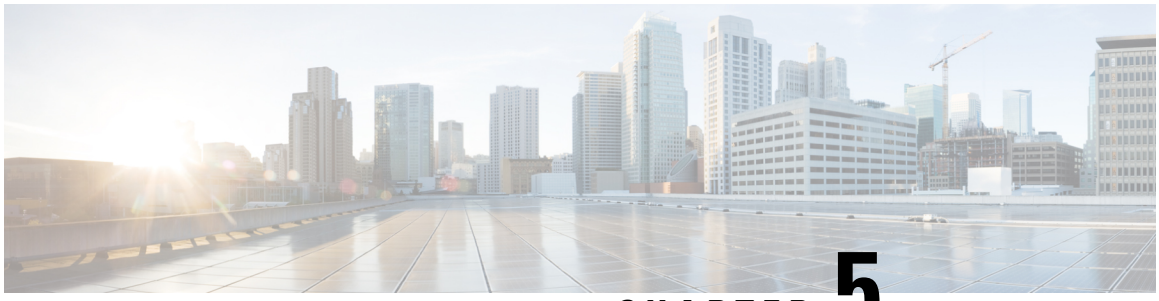
Step 14 Select the **Export CRD to Golden Repository** checkbox and specify the SVN host name in the text field.

Note
For CPC Converged Policy, the SVN host name value is **svn**.

Step 15 Click **+** to add the entry, then click **Export**.

A success message confirms that the CRD data has been exported to the golden repository.

The Policy Builder policy data and CRD data are restored from backup. The CPC Converged Policy system is returned to its pre-upgrade configuration state.



CHAPTER 5

Persistent storage for policy configuration

Persistent storage for policy configuration

Persistent storage is a storage solution that:

- retains configuration data after power or network resources are disconnected,
- The CPC provides various storage technologies for managing the configuration data. The CPC has pre-defined storage such as the OpenStack Cinder volume used for storing the CRD data. CPC optionally stores the CRD data in shared storage such as OpenStack Cinder (default) or local storage. In the case of deployment on bare metal servers, CPC uses the local storage class along with the default storage layer as the persistent storage.
- uses Kubernetes Persistent Volume (PV) and Persistent Volume Claim (PVC) abstractions to decouple storage from individual pods.

Restore Capability

The Subversion repository stores the policy-specific configuration data in the XMI format. This repository resides in an SVN pod. If the SVN pod is restarted, the repository experiences a data loss. In such scenarios, you must reimport the configuration files to the SVN pod.

A new restore mechanism is introduced to protect the configuration data and maintain its integrity when the SVN pod restarts.

How persistent storage works

Restore Capability

The restore capability maintains the continuity of the policy configuration files in conditions where the SVN pod is restarted.

The policy configuration files are in the XMI format. Each SVN repository contains XMI files that are represented in a configMap. The configMap is updated whenever a policy configuration is modified and committed into an SVN repository. When the SVN pod is restarted, it verifies if the configMap is available and the corresponding XMI files are loaded to the repository.

The restore capability is managed through these configMaps:

- Monitor-svn-configmap-pcf: Contains configuration data in key-value pairs that represent the repository name and policy hash.
- Policy-svn-persistence-configmap: Contains the configured value of the policy-configuration-restore configMap.

Configure persistent storage

Enable persistent volume claim

This section describes how to enable persistent volume claim to configure persistent storage.

Procedure

To enable persistent volume claim, use the following configuration:

Example:

```
config
  k8s
  use-volume-claims [ true | false ]
  end
```

Command parameters:

- **config** — Enters the configuration terminal.
- **k8s** — Enters the Kubernetes configuration mode.
- **use-volume-claims [true | false]** — Configures whether volume claims are used during NF deployment.
  - When set to **true**, the default storage class, such as OpenStack Cinder, is enabled.
  - When set to **false**, data is stored in memory and is lost on a pod restart.

Assign persistent storage

Assign the storage class so that CPC Converged Policy knows which persistent volume to use when storing Custom Resource Definition (CRD) data.

Before you begin

Ensure that **use-volume-claims** is set to **true** before assigning a storage volume. See the Enable persistent volume claim section.

Procedure

To assign the persistent storage volume class, enter the following configuration in the CPC Converged Policy CLI:

Example:

```
config
 db
   global-settings
      volume-storage-class [ default | local ]
      end
```

Command parameters:

- **volume-storage-class [default | local]** — Configures the storage that gets assigned as the persistent storage.
  - **default** — Uses the OpenStack Cinder shared storage class.
  - **db** — Enters the database configuration mode.
  - **local** — Uses the local storage class, which is required for bare metal server deployments.

Configure the restore capability

Procedure

To configure the restore capability, enter the following configuration in the Policy Ops Center console:

Example:

```
config
  engine engine_name
    pcf policy-configuration-restore [ true | false ]
    end
```

Command parameters:

- **engine** *engine_name* — Specifies the engine for which the restore capability must be configured.
- **pcf policy-configuration-restore [true | false]** — Configures the capability that restores the configMap when the SVN pod restarts. The default value is **true**.

Configure MongoDB storage size

Feature description

Table 6: Feature description

Feature Name	Release Information	Description
Configure MongoDB storage size	2026.03.0	This enhancement allows to define custom Persistent Volume Claim (PVC) storage sizes for individual MongoDB replica sets.

The MongoDB storage configuration feature enables you to define custom Persistent Volume (PV) storage sizes for individual MongoDB replica sets. Previously, the PCF enforced a fixed 5 Gi limit for all MongoDB PVCs, which prevented operators from scaling storage to meet specific Subscriber Profile Repository (SPR) data volumes and retention policies. This enhancement introduces a two-level configuration hierarchy, a global default and a per-replica-set override that allows you to optimize storage for specific environments, reducing under-provisioning risks and costs.

Configure the storage parameter

Before you begin

Ensure that you meet these requirements:

- Ensure you have administrative access to the PCF Ops-Center CLI.
- Verify that the required storage values are in specified range (5- 250).
- Understand that storage configuration changes apply only to new PVC provisioning. You cannot resize existing PVCs through this configuration.

Define custom persistent volume storage sizes for MongoDB replica sets.

Procedure

Step 1 Log in to the PCF Ops-Center CLI and enter `config` mode.

Step 2 Configure the global default storage size for all Session Control Database (SCDB) instances.

```
scdb storage <value>
```

Step 3 Configure the storage size for a specific MongoDB replica set. This step overrides the global default.

```
scdb replica-name <replica-name> storage <value>
```

Step 4 `commit` the configuration changes.

Example:

Sample configuration

```
scdb:
  storage: "5Gi"
  replicaName:
    - name: session
      storage: "10Gi"
    - name: balance
      storage: "8Gi"
```

Step 5 After the system reaches 100% operational status, confirm that Kubernetes PVC resources reflect the update.

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

Example:

```
NAME                          STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS   AGE
db-local-data-session-0       Bound    pvc-12345678-abcd-1234-5678-1234567890ab   10Gi       RWO
        local-storage  2m
db-local-data-balance-0       Bound    pvc-87654321-dcba-4321-8765-0987654321ba   8Gi        RWO
        local-storage  2m
```

Note
Interpretation of results:

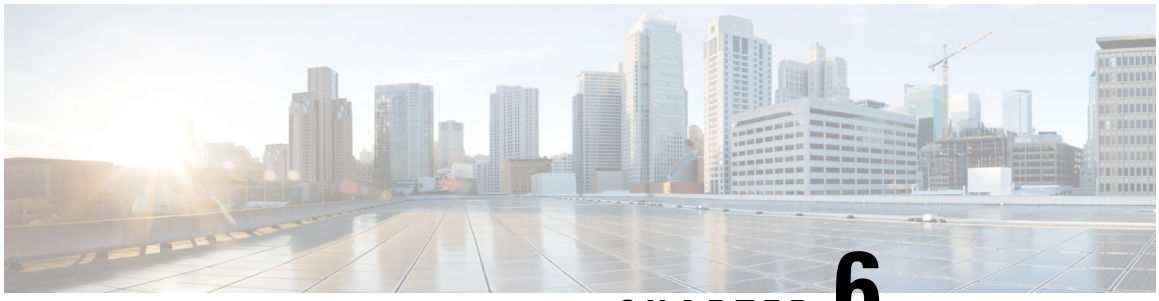
- **Success**: The system applies the new storage size to the PVC resources.
- **Syntax error**: Storage must be between 5Gi and 250 Gi.

Troubleshoot storage configuration

Use this section to troubleshoot and resolve common MongoDB storage configuration errors, including instances where the configuration fails to apply or the PVCs do not reflect the expected changes.

Table 7: Troubleshoot storage configuration

Issue	Corrective action
PVC resize failure	If the new size is not reflected by existing PVCs, check whether the Kubernetes cluster's storage class supports resizing.



CHAPTER 6

Manage Custom Reference Data

Custom Reference Data

Custom Reference Data (CRD) is reference data specific to a service provider—such as network names or cell site characteristics—that the policy engine requires to operate but does not use when evaluating policies. CRD includes:

- table-based storage that lets service providers create and manage custom data tables according to their requirements, and
- a pagination component that controls the number of data rows displayed per page, which you can change at any time; the configured value persists across the Central interface until you change it.

Note Restart all policy servers after modifying a CRD table schema—for example, after adding or removing a column.

Configuration support for importing CRD

This topic provides the steps required to import CRD when the CRD schema is modified. Importing of CRD involves these steps:

- Backing up the existing SVN repository
- Backing up the existing CRD
- Removing the existing CRD from MongoDB
- Importing and publishing the new CRD Schema
- Importing the new CRD table

Back up the SVN repository

Use this task to preserve the existing Policy Builder configuration and SVN data in a safe location before you modify the CRD schema.

Procedure

Step 1 Log in to the CPC Central GUI.

Step 2 On the **Converged Policy & Charging Central** page, navigate to **Policy Builder** and click the **Import/Export** link.

The **Import/Export** form opens.

Step 3 In the **Export** tab, select the **All data** option to configure the export type.

The export and import options are described in the table.

Table 8: Export and import options

Parameter	Description
All data	Exports service configuration with environment data, providing a complete backup of both service configuration and environment data.
Exclude Environment	Exports without environment data, allowing you to export configuration from a lab environment without overwriting environment-specific data in the target system.
Only Environment	Exports only environment data, providing a way to back up system-specific environment information.
Export URL	The URL that you can access from Policy Builder or view directly in Subversion.
Export File Prefix	A name prefix for the export file. **Note** The exported filename automatically includes the date and time of the export.

Step 4 To export in compressed format, select the **Use 'zip' file extension** check box.

Step 5 Click **Export**.

Step 6 Navigate to the exported file and save it to your local machine. Include the cluster name and date in the filename.

Back up the existing CRD

Use this task to preserve existing CRD data in a safe location before you modify the CRD schema. This backup allows you to restore the data if the schema update fails.

Procedure

Step 1 Log in to the CPC Central GUI.

Step 2 On the **Converged Policy & Charging Central** page, navigate to **Custom Reference Data** and click the **Custom Reference Data** link.

The **Import/Export CRD data** form opens.

Step 3 Under **Export Custom Reference Data**, configure the export options as needed.

Table 9: Export Custom Reference Data options

Option	Description
Use 'zip' file extension	Enables easier viewing of exported contents for advanced users.
Export CRD to Golden Repository	When the system is in a bad state, the CRD cache is rebuilt using the golden CRD data.

Step 4 Click **Export**.

Remove the existing CRD from MongoDB

Use this task to clear the affected CRD table data from MongoDB so the updated schema can be imported without conflicts caused by the old data structure.

Before you begin

Complete the CRD backup tasks before performing this task. Removing data without a backup may result in unrecoverable data loss.

Procedure

Step 1 Log in to the admin-db pod that contains the CRD (**cust_ref_data**) database.

Step 2 Access the **cust_ref_data** database.

Example:

```
use cust_ref_data
```

Step 3 Delete the data from the CRD table or tables that have schema changes.

Example:

```
db.table_name.remove({})
```

Replace **`table_name`** with the name of each CRD table you want to clear. Repeat this command for each affected table.

Step 4 Exit the admin-db pod.

MongoDB replica sets

MongoDB replica set configurations support both IPv4 and Internet Protocol version 6 (IPv6) addresses for replica member hosts, which ensures that initialization, replication, and initial synchronization work correctly regardless of the IP version in use. This dual support allows flexible network configurations and simplifies transitions to IPv6 while maintaining compatibility with existing IPv4 infrastructure.

Note Configure each replica subscriber with either IPv4 or IPv6 addresses. Do not mix IP versions within a single subscriber configuration.

Configure IPv6 in MongoDB replica sets

Use this task to configure the database, replica set name, port, interface, memory limits, node labels, and member hosts when deploying MongoDB replica sets that use IPv6 addresses.

Procedure

Step 1 In Ops-Center, configure the database and replica set initialization parameters.

Example:

```
Database: db scdb
Replica Set Name: replica-name mongo-spr1
Port: 27018
Interface: interface db1.mdb.2564
Resource Memory Limit: 112000
```

Step 2 Configure the replica set labels and member hosts.

Example:

```
Replica Set Label Key: smi.cisco.com/node-type-5
Replica Set Label Value: db-spr

Member Configurations:
Member: sprdb-rs1-arbiter
Host: 2001:DB8::8
Arbiter: true
Site: remote

Member: sprdb-rs1-s1-m1
Host: 2001:DB8::9
Arbiter: false
Priority: 166
Site: remote

Member: sprdb-rs1-s1-m2
Host: 2001:DB8::10
Arbiter: false
Priority: 155
```

```
Site: remote

Member: sprdb-rs1-s2-m1
Host: 2001:DB8::11
Arbiter: false
Priority: 66
Site: local

Member: sprdb-rs1-s2-m2
Host: 2001:DB8::12
Arbiter: false
Priority: 55
Site: local
```

Importing CRD when the schema changes

Perform this process when a CRD table schema is modified—for example, when a column is added or removed. Skipping or reordering the stages can cause data loss or import failures.

Summary

The importing CRD process involves the operator working across the CPC Central GUI and the MongoDB admin-db pod to replace the old schema and data with the new version.

Workflow

These are the stages of importing CRD when the schema changes:

1. Stage 1 — Back up the SVN repository: Export the current Policy Builder configuration from the CPC Central GUI to preserve a fallback copy.
2. Stage 2 — Back up the existing CRD: Export the CRD data to a file so you can restore it if needed.
3. Stage 3 — Remove the existing CRD from MongoDB: Delete the affected CRD table data from the MongoDB **cust_ref_data** database to clear the stale schema.
4. Stage 4 — Import and publish the new CRD schema: Import the updated schema file into Policy Builder and publish it to the runtime environment.
5. Stage 5 — Import the new CRD table: Load the new CRD data archive into CPC Central and verify successful import.

Result

After completing all five stages, the CRD tables reflect the new schema and the system operates with the updated reference data.

CPC table evaluation optimization

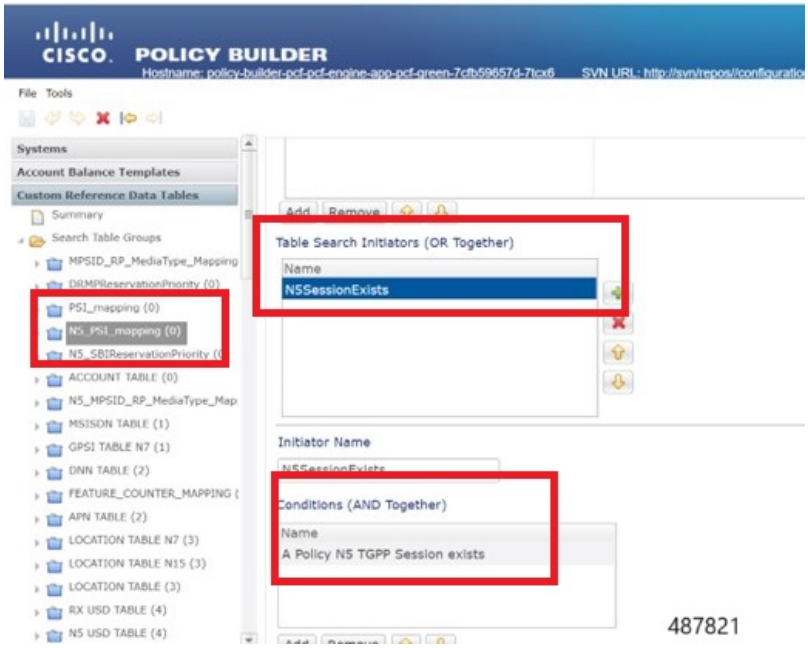
The CPC currently evaluates all the configured CRD tables for every incoming message. This comprehensive evaluation complicates troubleshooting because it is difficult to determine the source of specific values. Therefore, CPC allows the use of Table Search Initiator conditions, which activate only the required tables, depending on specific conditions.

You can add initiator conditions for N5, Rx, and non-N5/Rx tables.

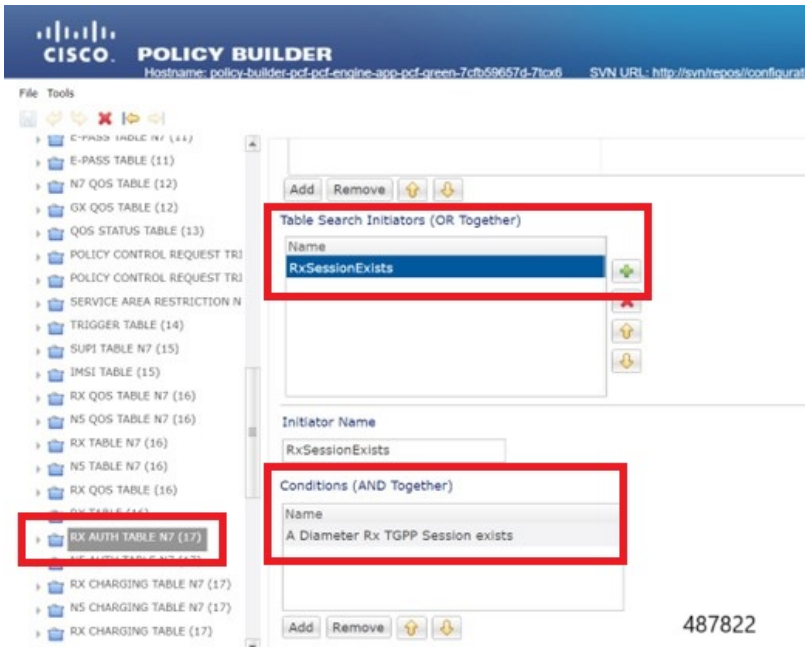
Initiator Conditions

The Table Search Initiator conditions are included for the respective tables for evaluation:

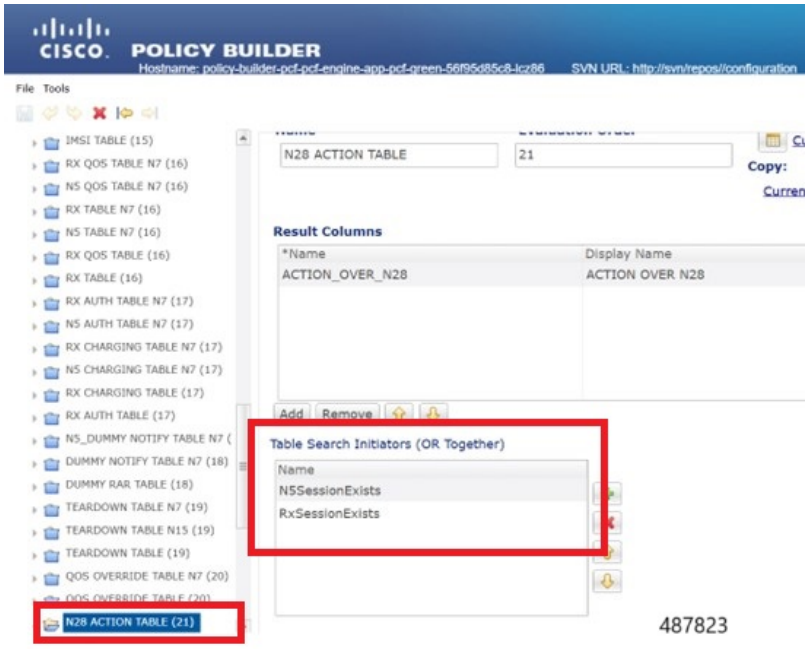
- For N5 Tables, use the **A Policy N5 TGPP Session exists** condition.

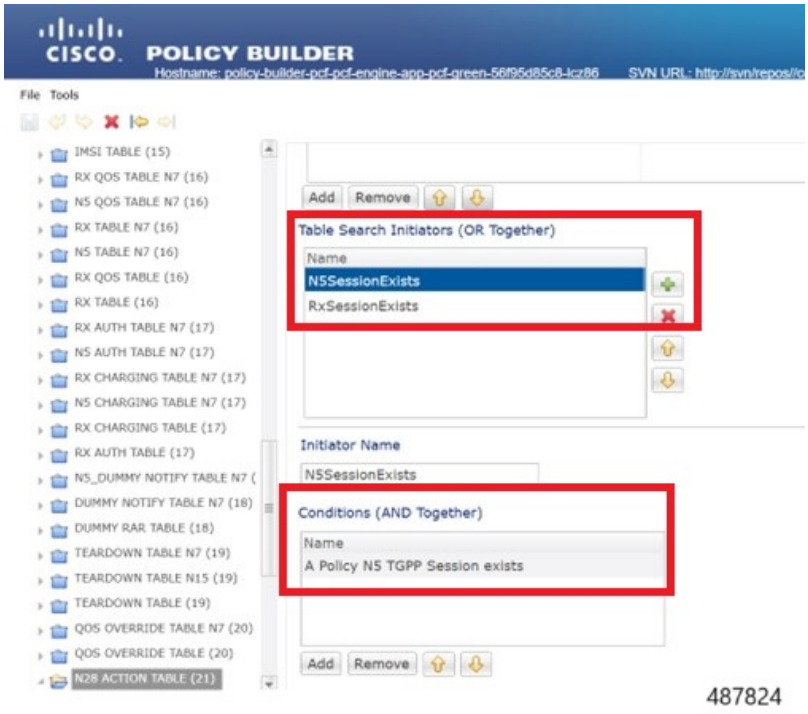


- For Rx Tables, use the **A Diameter Rx TGPP Session exists** condition.



- Use a **A Policy N5 TGPP Session exists** or **A Diameter Rx TGPP Session exists** condition for the tables to avoid evaluation for messages other than N5 or Rx message:





Import the new CRD table

Use this task after publishing the new CRD schema to load the updated CRD table data into the system.

Before you begin

Ensure the CRD data archive is saved as a **.crd** or **.zip** file before you begin.

Procedure

Step 1 Log in to CPC Converged Policy Central.

Step 2 Click **Custom Reference Data**.

Step 3 Click **Import/Export CRD Data**.

Step 4 Under **Export Custom Reference Data**, configure export options as required.

- Select the **Use 'zip' file extension** check box to enable easier viewing of exported contents for advanced users.
- Select the **Export CRD to Golden Repository** check box to export CRD to the golden repository, which is used to restore **cust_ref_data** in error scenarios. A text box appears for you to enter the golden repository hostname.

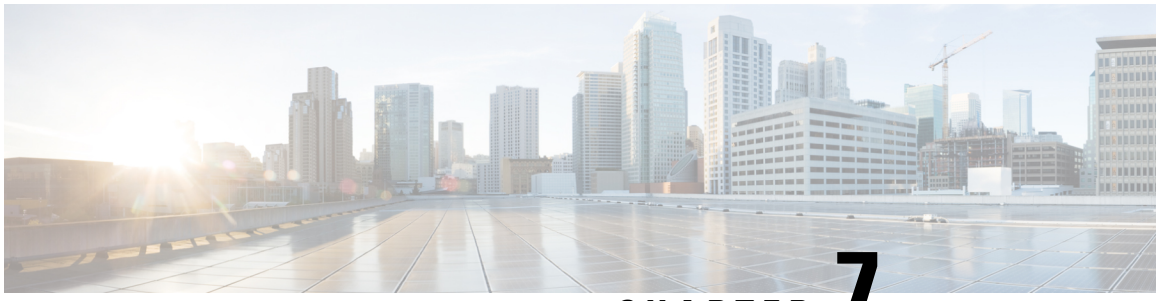
Step 5 Add a valid SVN server hostname or IP address to push the CRD to the repository.

To add multiple hostnames or IP addresses, click the plus (+) icon.

Step 6 Click **Export**.

To verify the export to the golden repository was successful:

1. Log in to CPC Converged Policy Central.
2. Click **Custom Reference Data**.
3. Click **Import/Export CRD Data**.
4. In **Import Custom Reference Data**, click **Field to Import field** and browse for the CRD archive.
5. Click **Import** to import the CRD data.
6. Verify that a "Data imported" message appears on the CPC Converged Policy Central GUI.
7. Review the crd-api pod logs for any exception or error related to a duplicate key or duplicate index. If no errors appear, the CRD is successfully imported.



CHAPTER 7

Policy Builder Overview

Converged policy and charging

CPC allows service providers to create policies that are customized to their particular business requirements through the use of the CPC Policy Builder, a web-based tool with a graphical user interface (GUI) that allows for rapid development of innovative new services.

The Policy Builder GUI supports both configuration of the overall CPC cluster of virtual machines (VMs) as well as the configuration of services and advanced policy rules. These sections introduces the main aspects of the PB GUI as laid out in three tabs on the upper right of the interface: Reference Data, Services and Policies.

Reference data

A reference data item is a system configuration element that

- provides access for configuring various aspects of the system to prepare it for operation
- supplies settings and parameters referenced by policy rules across multiple services, and
- enables reuse of configuration options such as account balances and notifications by different services as needed.

Reference data configuration options

The Reference Data tab contains static system, network, and template definitions. It is not directly related to policy, services, or use cases, but defines the reference points for the following types of information:

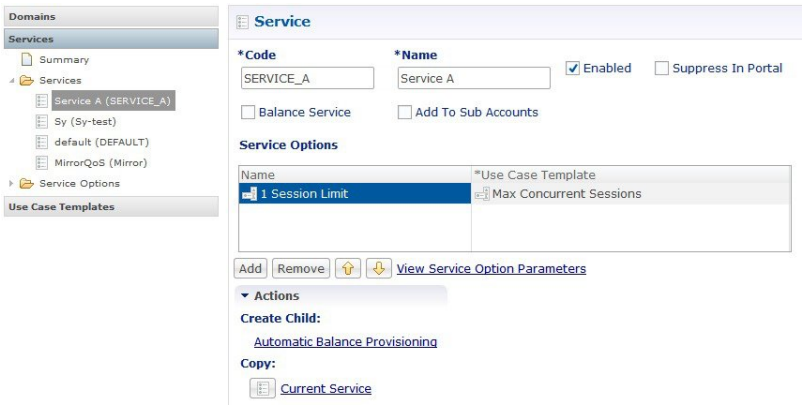
- Systems, cluster, and instance data
- Jdbc query string definitions
- Balance and quota definitions
- Diameter configuration, and CPC (Rest API - 5G) configuration
- Query strings
- Custom reference data tables (custom look up tables such as apn names)
- Notification addresses and text templates
- Policy reporting criteria
- Subscriber data repositories

Services

The creation of a new service begins with creating a Use Case Template (UCT) for the service. UCTs consist of Service Configurations that are specific to the service being created. For example, a Service Configuration might set up a Gx Rule or Basic QoS. The UCT is also used to configure Use Case Initiators (UCI), which are instructions describing when a specific Service Configuration should be in effect. An example of a UCI is only send this Gx Rule when the account balance is depleted. Multiple UCIs can be configured for each Service Configuration. This enables complex logic that determines when the configuration is active or inactive.

Once a UCT and associated UCIs are defined, it becomes the basis for Service Options, which are specific instances of the UCT that are populated with data specific to the service. Multiple Service Options can be created from a single UCT; for example, a UCT that provides for passing QoS parameters can be reused with different QoS values for different customers. Multiple Service Options can be layered to create the end Service.

Figure 9: Services tab



The Domains panel within the Services tab handles the initial interaction of the client device with the policy engine, and covers tasks including client authentication, default provisioning of unknown clients and qualifying a client for particular system defaults and services.

Policies

Within the various Extension Points are Policies that define Conditions (events and data from the policy flow and external systems) that can then trigger Actions (manipulation of data and communication back to external systems).

Note that the configuration of services for most deployments will be handled through use of the Reference Data and Services tabs; advanced policies as defined on the Policies tab and discussed above are only required for complex deployments. It is recommended that only experienced users access the Policies tab as errors in custom policies can have negative impact on the operation of the system. Detailed discussion of custom policies is outside of the scope of this document.

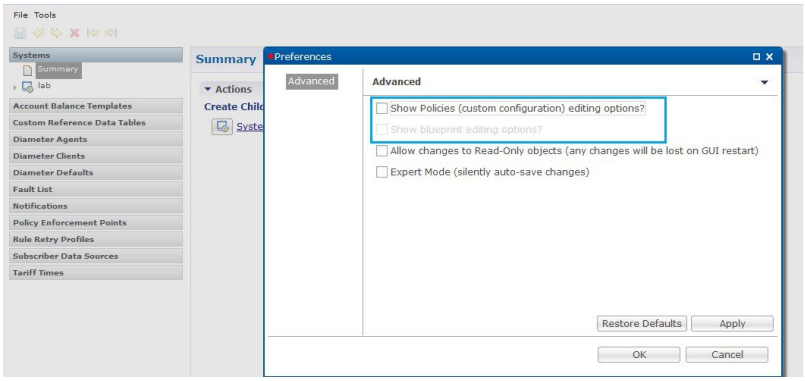
By default, the Policies and Blueprint tabs are disabled.

Important The Policy Builder offers the Blueprint section under **Policies** tab to enable Cisco recommended changes to the Policy Engine. Changes made without Cisco guidance are not supported and can result in poor performance, platform instability, or reduced capacity.

Enabling Policies tab

In Policy Builder, **Tools** > **Preferences** to open **Preferences** pop-up window.

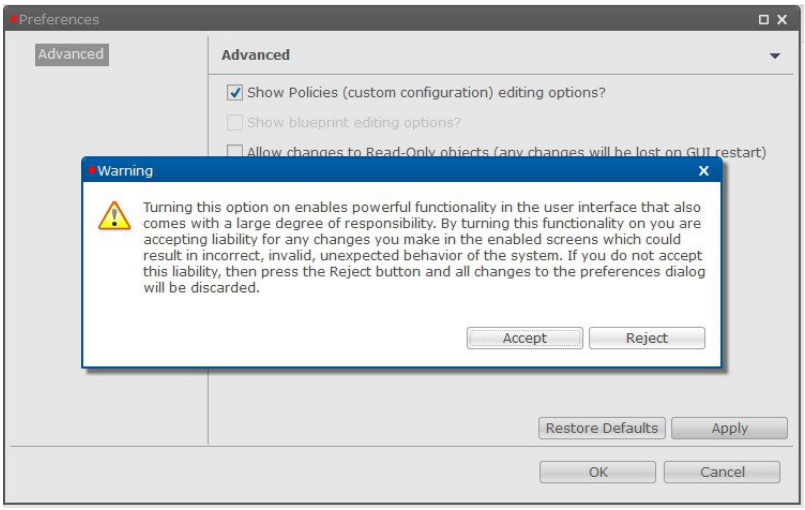
Figure 10: Preferences - Policies



Select Show Policies (custom configuration) editing options? and click **Apply**.

Warning pop-up dialog box opens up.

Figure 11: Warning



Click **Accept** so that **POLICIES** tab is visible in Policy Builder.

Enabling BLUEPRINTS tab

For **BLUEPRINTS** tab to be visible in Policy Builder, Show Policies (custom configuration) editing options? must be checked.

Select Show blueprint editing options? and click **Apply**.

Figure 12: Preferences - Blueprints 1

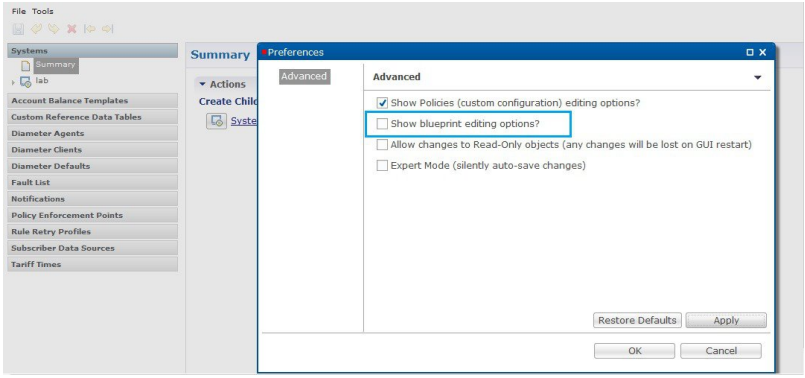
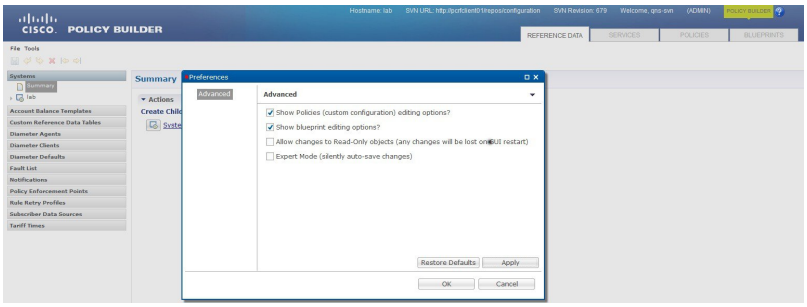


Figure 13: Preferences - Blueprints 2



Warning pop-up dialog box opens up.

Click **Accept** so that **BLUEPRINTS** tab is visible in Policy Builder.

Note In case the **POLICIES** checkbox is unchecked while **BLUEPRINTS** checkbox is checked, the **BLUEPRINTS** checkbox is unchecked forcefully and the **BLUEPRINTS** tab is not visible.

Summary of Policy tab capabilities

This reference provides an overview of the Policy tab capabilities, focusing on conditional rules, action triggers, and session data evaluation for policy decisions and service data return.

- Conditional rules within specified Extension Points (Condition/Action)
- Trigger specific actions from an extensive catalog of Use Case Initiators
- Evaluate and manipulate session data as part of making policy decisions and returning services data to downstream systems

Advantages

- Allows for handling complex policy situations without writing custom code.
- Support for custom or unusual business rules,

Considerations for building custom policies

- Building custom policies requires a deep understanding of the call flow and underlying CPC platform.
- Due to the flexibility of the Policy Builder, it is possible to create conflicting policies that can have a negative impact on system performance.

Accessing the Policy Builder

The Policy Builder is the web-based client interface for the configuration of policies to the Converged Policy and Charging. Initial accounts are created during the software installation with the default CPC install username as **`qns-svn`** and password as **`cisco123`**.

URL to access Policy Builder Interface:

- For HA: https://<lbvip01>:7443/pb

CPC enables users to be aware of its current privileges while accessing Policy Builder as described below:

- If a user has read-write privilege then ADMIN is displayed adjacent to user name in the GUI.
- If a user has read-only privilege then READONLY is displayed adjacent to user name in the GUI.

The hostname is displayed in the login dialog box and system banner to differentiate between open windows while performing any operation of the CPC system. It indicates which system is being modified and prevents any errors or misconfigurations.

The hostname is displayed when the parameter **-Dhostname=lab** is configured in **pb/qns.conf** files. If it is not configured in the qns.conf file, it is displayed as a result of the command "hostname" on the server.

The hostname is displayed in the login panel only when the following argument is set to true:

-DshowSitenameLogin

Enable TACACS+ authentication for Policy Builder by enabling PAM authentication (set -DdisablePAMAuthentication to false) and enabling TACACS+ along with `tacacs_on_ui` flag set to true in Configuration.csv file.

Policy Builder Field Value Validation

When you start the Policy Builder, the last recorded valid value defined in the eCore is set as the default value. If the value is not set in the eCore, the valid value is taken based on the eCore data type. For numeric data types, the Policy Builder displays 0 as a default value. For string data types, the Policy Builder displays null (empty) as a default value.

The Policy Builder validates the value that is configured in the field.

When you enter a value in the field and the value passes validation, the newly entered value is recorded as the last valid value. If the validation fails, the last recorded valid value is reverted.

Tarrif of the day

Feature description

Table 10: Feature description

Feature Name	Release Information	Description
Tariff of the day	2026.03.0	The Tariff of the Day feature allows service providers to define and apply variable billing rates and different data rules for specific time intervals. This feature supports dynamic quota management and precise cost control. Service providers benefit from automated, time-based rate switching, which ensures subscribers are billed accurately based on pre-defined tariff periods.

Tariff Times

Tariff Times refers to the PCF terminology that defines rate structures and their applicable time periods. To determine the current TariffSwitchTime, Balance uses the current time, based on the time zone specified in the tariff time reference data configuration. It then checks each switch time from top to bottom to determine whether the current time matches a TariffSwitchTime. The Policy Control uses the first tariff switch time that matches, including any associated valid dates or a holiday date. The tariff switch time marks the end of the current tariff period. The system calculates the next tariff switch time by adding one second to the end of the current tariff switch time and then searching each tariff switch time from top to bottom to find the next match.

Rates

Rates allow service providers to change how quota is billed to customers. Typically, there is a one-to-one relationship: if a customer uses 1 MB, they are charged $1 for that usage. Service providers can modify the rate to alter the cost subscribers pay for their quota usage. For example, with a rate of 0.25, the quota is billed at 1 MB for every 4 MB used. With a rate of 2, quota usage is billed at 2 MB for every 1 MB the customer uses.

The rate is specified when the system makes a reservation. The default rate is 1.

Tariff Times configuration- Rate specific usecase

This section explains how to set up rates for components that use Autowire Balance. Autowire Balance is the default blueprint for Balance. You must configure it in the Policy Builder for use in the base system setup. Autowire Balance extends the main system blueprint.

Complete these steps to configure tariff times and balance rates in the Policy Builder:

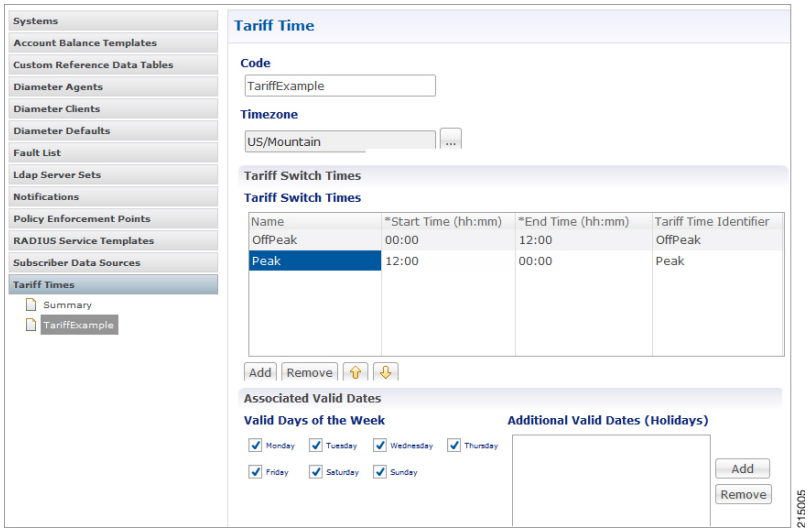
Procedure

Step 1 Configure tariff times:

a) Navigate to **Reference Data** > **Tariff Times**.
b) Create a new child entry and set the time zone if required.
c) Define rows that cover a full 24-hour period.

Note
A start time of 00:00 (midnight) is inclusive, while an end time of 00:00 is exclusive. Using 00:00 as the end time ensures the system accounts for all 86,400 seconds (24 hours) in a day.

Figure 14: Tarrif Time



Step 2 Configure balance rate settings:

a) Navigate to **Service** > **Use Case Templates**.

b) Click **Add** in **Service Configuration** and select **BalanceRateConfiguration**.

c) Choose an **Account Balance Template**.

Step 3 Configure rate parameters:

a) In the **Rates List**, select a **Tariff Switch Time (Key)**.

b) Modify the rate as needed.

c) Click **Add Child** to add additional rate options.

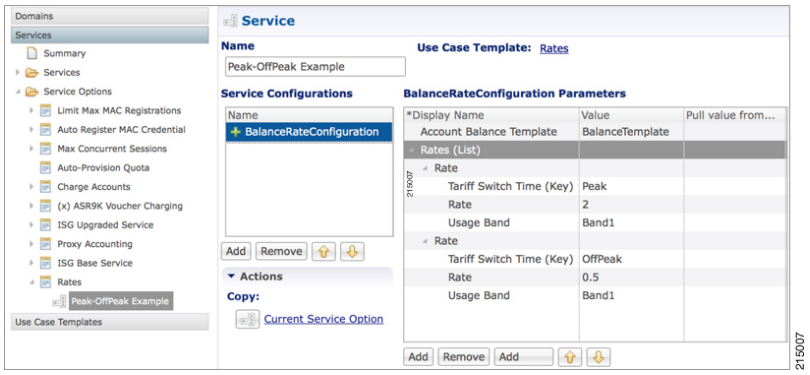
Step 4 Configure service options:

a) Navigate to **Services** > **Service Options** > **Rates**.

b) Click **Create Child Service Option**.

c) Click **Add** in **Service Configuration** and select **BalanceRateConfiguration**.

Figure 15: Service Option



Step 5 Choose **File** > **Publish to Runtime**.

Tariff Times configuration - TimeOfDay (ToD) usecase

By using Tariff Time configs, CPC also can install/uninstall the TimeOfDay Rules for the subscriber [ex. Unlimited Data Usage Rules during night or weekend] using Tariff Switch Times configuration. Sample configuration given below.

Complete these steps to configure tariff times and TimeofDay in the Policy Builder:

Procedure

Step 1 Configure tariff times:

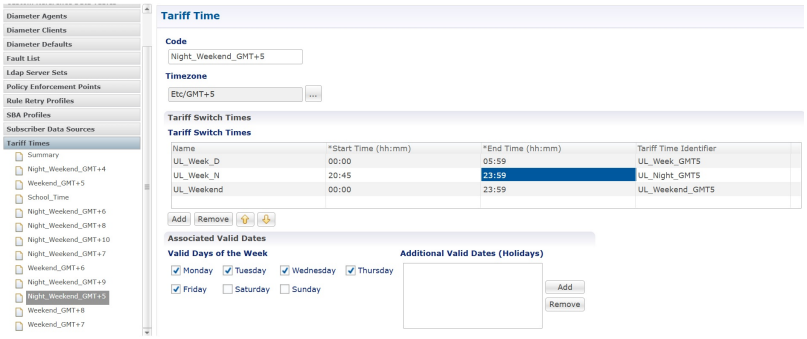
a) Navigate to **Reference Data** > **Tariff Times**.

b) Create a new child entry and set the time zone if required.

c) Define rows that cover a full 24-hour period.

Note
A start time of 00:00 (midnight) is inclusive, while an end time of 00:00 is exclusive. Using 00:00 as the end time ensures the system accounts for all 86,400 seconds (24 hours) in a day.

Figure 16: Tarrif Time



Step 2 Configure TimeofDay rate settings:

a) Navigate to **Service** > **Use Case Templates**.

b) Click **Add** in **Service Configuration** and select **TimeofDay**.

Step 3 Select a **Tariff Switch Time (Key)**.

a) Navigate to **Services** > **Service Options** > **TimeofDay**, click **Create Child Service Option**, and select **TimeofDay** in the **Service Configuration**.

b) Click **Add Child** to add one more Use Case Template to install Rule based on condition.

c) Add condition to match **Tariff Time Code**which is mentioned in the Tariff Times Reference Data.

Step 4 Configure service options:

a) Navigate to **Services** > **Service Options** > **TarrifTime** Service option.

b) Click **Create Child Service Option**.

c) Click **Add** in **Service Configuration** and select **TimeofDay Configuration**.

Figure 17: Service Option

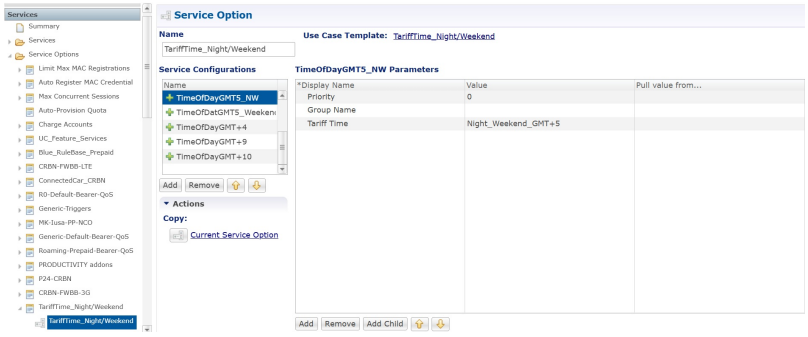
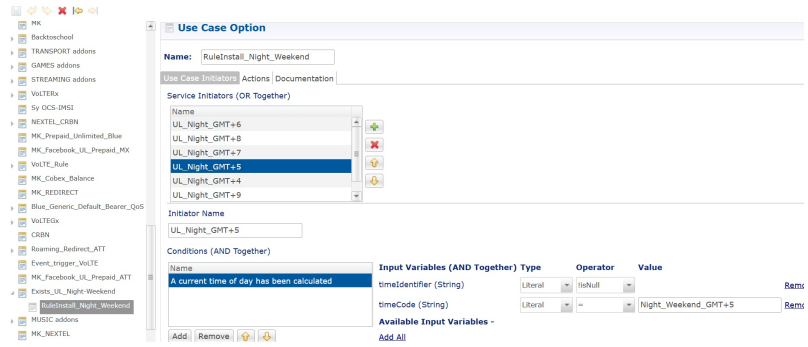


Figure 18: Use Case Option



Step 5 Choose **File** > **Publish to Runtime**.

Edge cases

It is strongly recommended that Tariff Switch Times cover all times during a 24 hour period and do not have gaps. Some customers use a default Tariff Switch Time entry that covers all the other times that have not been specifically defined.

Tariff Times are not allowed to cross over the midnight boundary for a given day. In practice, this means that often 2 or more tariff switch times must be created to cover a single logical period. For Example:

Edge case examples

Table 11: Edge case examples

Name	Start Time	End Time	Tariff Time Identifier
Nights Before Midnight	17:00	00:00	NIGHT
Nights After Midnight	00:00	07:00	NIGHT
Days	07:00	17:00	DAY

If a Tariff Switch occurs during a daylight savings time forward switch (i.e. between 2:00am and 3:00am during 'Spring Forward' in March), an error will occur in processing during that time since that hour is lost. Therefore, it is recommended that switch times NOT occur during these times on those days.

Gx based quota management

Overview

CPC supports Usage-Monitoring over Diameter Gx interface with different Balance Code, Dosage and monitoring level. Usage monitoring key Identifies the usage monitoring control instance.

Policy Builder configuration

Procedure

Step 1 Login to CPC Policy Builder.

Step 2 Configure the Account Balance Templates in the **Reference Data** tab. For more information, see Account Balance Templates.

Step 3 Select the `Services` tab, and then click `Use Case Templates` > `Summary`.

Step 4 Click **Use Case Template** link from the right side under **Create Child** to create a use case template.

a. Enter the name for use case template. For example, name the new template as *UsageMonitoringKey*.

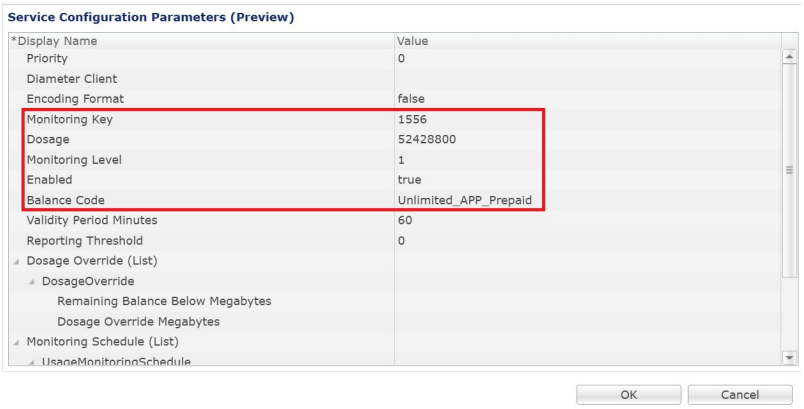
b. Click **Add** under **Service Configurations**.

Select Service Configuration dialog box opens, and all of the service configuration objects that are available are listed.

c. Scroll down to the **gx** area in the list of service configuration objects, and select the required usage monitoring object. For example, **UsageMonitoringKey**.

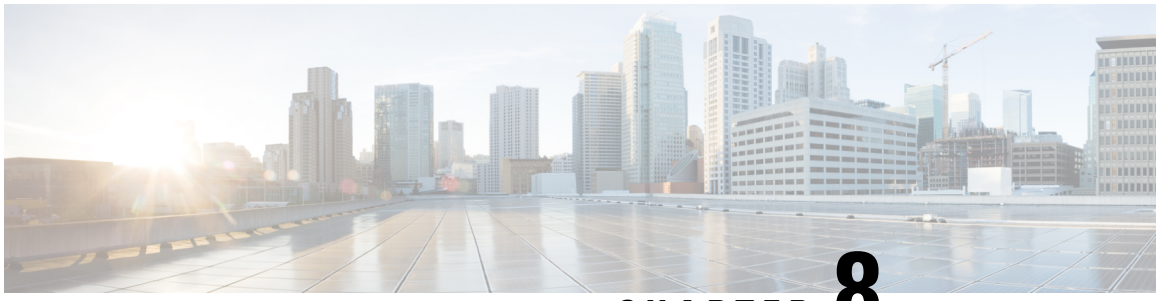
d. Select **Balance Code** in the UsageMonitoringKey parameter. Add other parametes as per the reference image. Map this Balance Code to the previously created **Account Balance Template**.

Figure 19: Usage monitoring schedule



Step 5 In the left pane of the Services tab, click **Services** > **Service Options** to create a service option and add the *UsageMonitoringKey* use case template.

Step 6 In the left pane of the **Services** tab, click **Services** > **Services** to create a service and add the configured *UsageMonitoringKey* use case template.



CHAPTER 8

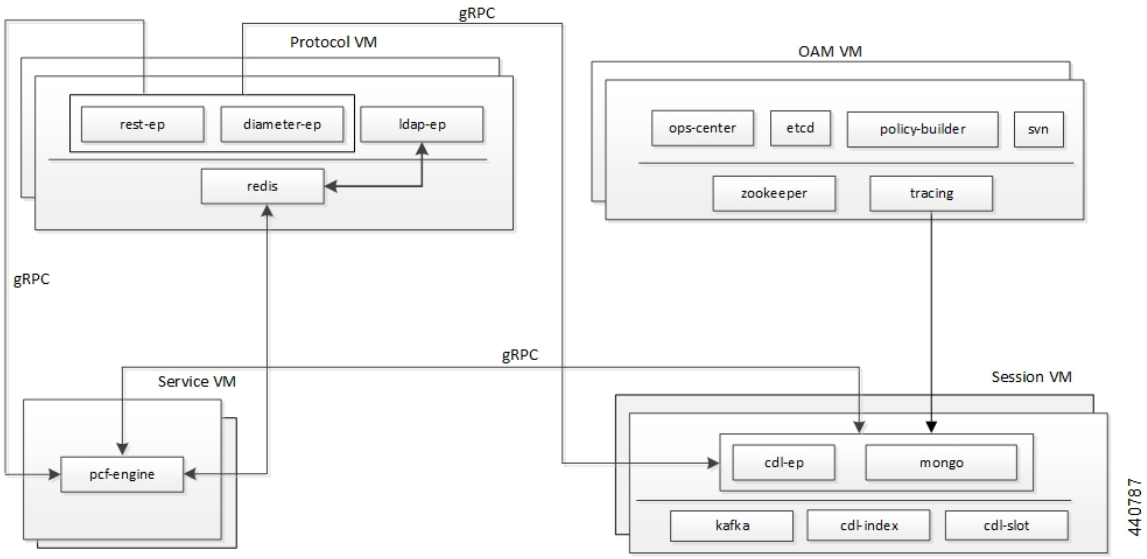
Pods and services

CPC pods and services overview

Depending on your deployment environment, CPC deploys the pods on the virtual machines that you have configured. Pods operate through the services that are responsible for the intrapod communications. If the machine hosting the pods fail or experiences network disruption, the pods are terminated or deleted. However, this situation is transient and CPC spins new pods to replace the invalid pods.

This workflow provides a high-level visibility into the host machines, and the associated pods and services. It also represents how the pods interact with each other. The representation might defer based on your deployment infrastructure.

Figure 20: Communication Workflow of Pods



The Protocol VM hosts the rest-ep, diameter-ep, and ldap-ep pod that governs the ingress (incoming) and egress (outgoing) traffic on the interfaces. The pods responsible for the operations and management processes reside in the OAM VM and, the Service VM hosts the pcf-engine. The session VMs hosts the pods that operate as the databases to store the data accessed by the pods. The illustration also depicts the services which the pods use to channel the interactions. The pods communicate over the gRPC interface.

Note Typically, multiple instances of the Protocol and OAM VMs are created to ensure resiliency.

Kubernetes deployment includes the kubectl command-line tool to manage the resources in the cluster. You can manage the pods, nodes, and services using the CLI.

For performing the maintenance activities, you can use the **kubectl drain** command to withdraw a node voluntarily. This command prepares the node by evicting or assigning the associated pods to another node with sufficient resources. You can run the **kubectl drain** on individual or multiple nodes concurrently.

Pods

Kubernetes deploys one or multiple pods on a single node which can be a physical or virtual machine. Each pod has a discrete identity with an internal IP address and port space. However, the containers within a pod can share the storage and network resources.

This table lists the pod names and the hosts on which they are deployed depending on the labels that you assign.

Table 12: CPC Pods

Pod Name	Description	Host Name
admin-db	Acts as the MongoDB router pod for the Admin database.	Session
api-pcf-ops-center	Functions as the confD API pod for the CPC Ops Center.	OAM
cdl-ep-session-c1	Provides an interface to the CDL. **Note** Configuration changes to the CDL endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Session
cdl-index-session	Preserves mapping information of the keys to the session pods.	Session
cdl-slot-session-c1	Operates as the CDL Session pod to store the session data.	Session
cps-license-manager	Acts as the CPC License Manager.	OAM

Pod Name	Description	Host Name
crd-api-pcf-pcf-engine-app-pcf-<*n*>-mjgxp	Hosts the CRD APIs.	Protocol
db-admin	Acts as the replica set pod for the Admin database.	Session
db-admin-config	Acts as the replica set pod that stores the Admin database configuration.	Session
db-spr-config	Operates as the replica set pod that stores the SPR database configuration.	Session
db-spr1	Functions as the replica set pod that preserves the SPR database.	Session
diameter-ep-rx-rx	Contains the Diameter stack details and acts as the endpoint. **Note** Configuration changes to the diameter endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
documentation	Contains the documentation.	OAM
etcd-pcf-etcd-cluster	Hosts the etc-d for the CPC application.	OAM
grafana-dashboard-cdl	Contains the Grafana metrics for CDL.	OAM
grafana-dashboard-pcf	Contains the Grafana metrics for CPC.	OAM
kafka	Hosts the Kafka details for the CDL replication.	Protocol
ldap-ep	Operates as an LDAP client to establish communication with an external LDAP server. **Note** Configuration changes to the LDAP endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
network-query	Operates as the utility pod to determine the route IP for the Diameter outbound messages.	OAM
ops-center-pcf-ops-center	Acts as the CPC Ops Center.	OAM
patch-server-pcf-cnat-cps-infrastructure	Operates as the utility pod for patching the CPC JAR files.	OAM

Pod Name	Description	Host Name
pcf-day0-config-pcf-pcf -engine-<*n*>-rchg2	Dedicated for performing the Day-0 configuration for CPC.	OAM
pcf-engine-pcf-pcf-engine-app-pcf	Operates as the CPC Engine. **Note** Configuration changes to the CPC Engine endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Service
pcf-rest-ep	Operates as a REST endpoint for CPC. **Note** Configuration changes to the REST endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
policy-builder-pcf-pcf-engine-app	Operates as the Policy Builder for CPC.	OAM
redis-keystore	Operates as the REDIS Index.	Protocol
redis-queue	Operates as the REDIS IPC.	Protocol
rs-controller-admin	Responsible for the replication controller for Admin database.	Session
rs-controller-admin-config	Operates as a replication controller for the Admin database configuration.	Session
rs-controller-spr-config	Operates as a replication controller for SPR database configuration.	Session
rs-controller-spr1	Operates as a replication controller for the SPR database.	Session
smart-agent-pcf-ops-center	Operates as the utility pod for the CPC Ops Center.	OAM
svn	Stores all the CPC XMI configuration files.	OAM
svn-ldap	Stores the LDAP endpoint configuration which is configured through the ops-center.	Protocol
swift-pcf-ops-center	Operates as the utility pod for the CPC Ops Center.	OAM
traceid-pcf-pcf-engine	Stores the subscriber tracing details.	OAM
zookeeper	Assigned for the Zookeeper.	OAM

Services

This table describes the CPC services and the pod on which they run.

Table 13: CPC Services and Pods

Service Name	Pod Name	Description
admin-db	admin-db-0	Serves to process the MongoDB-specific router messages.
cps-diameter-inbound-rx-rx-rx	cps-diameter-ep	Transmits the Rx messages to the Diameter endpoint. You can set an external IP address for the service.
crd-api-pcf-pcf-engine-app-pcf	crd-api	Processes the CRD API calls.
datastore-ep	datastore-ep	Processes the CDL endpoint calls.
datastore-ep-session	ngn-datastore-ep	Responsible for the CDL session.
datastore-notification-ep	pcf-engine	Responsible for sending the notifications from the CDL to the engine.
diameter-engine	pcf-engine	Acts as the Diameter endpoint to pcf-engine.
documentation	documentation	Processes the documentation API calls.
etcd	pcf-etcd-cluster	Processes the etc-d API.
etcd-pcf-etcd-cluster-<n>	pcf-etcd-cluster	Processes the etc-d stateful sets.
grafana-dashboard-cdl	grafana-dashboard-cdl	Responsible for managing the Grafana dashboard for inputs from the CDL.
grafana-dashboard-pcf	grafana-dashboard-pcf	Manages the Grafana dashboard for CPC.
helm-api-pcf-ops-center	helm-api	Manages the Ops Center API.
kafka	kafka	Processes the Kafka messages.
mongo-admin-<n>	db-admin-0	Responsible for the Admin database stateful sets.
mongo-admin-config-<n>	db-admin-config-0	Responsible for the Admin database configuration stateful sets.
mongo-spr-config-<n>	db-spr-config-0	Responsible for the SPR database configuration stateful sets.
mongo-spr1-<n>	db-spr1-0	Responsible for the SPR database stateful sets
ops-center-pcf-ops-center	ops-center	Manages the CPC Ops Center.

Service Name	Pod Name	Description
patch-server-pcf-cnat-cps-infrastructure	patch-server	Maintains the patch repository.
pcf-day0-config-pcf-pcf-engine-app-pcf	pcf-day0-config	Manages the Day-0 configuration.
pcf-engine	pcf-engine	Processes the API calls to pcf-engine.
pcf-rest-ep	pcf-rest-ep	Acts as the http2 request/response to the REST endpoint. You can set an external IP address for the service.
policy-builder-pcf-pcf-engine-app-pcf	policy-builder	Manages the Policy Builder's request/response messages.
redis-keystore-<n>	redis-keystore-0	Manages the REDIS keystore stateful set.
redis-queue-<n>	redis-queue-0	Processes the REDIS queue stateful set.
rs-admin	replica-set admin	Manages the replica set for Admin database.
rs-admin-config	replica-set admin-config	Manages the replica set for the Admin database configuration.
rs-spr-config	replica-set spr-conifg	Manages the replica set for the SPR configuration.
rs-spr1	replica-set sp1	Manages the replica set for the SPR database.
smart-agent-pcf-ops-center	smart-agent-pcf-ops-center	Responsible for the Ops Center API.
svn	cps-subversion	Responsible for the SVN API calls.
swift-pcf-ops-center	swift-pcf-ops-center	Responsible for the Ops Center API.

Limitations

When removing a node using the `kubectl` drain command, the pods managing the inbound traffic such as `pcf-rest-ep`, `pcf-ldapserver-ep`, and `diameter-ep-rx-protocol` cannot be assigned to another node. The workload of these pods cannot be scheduled to another node since the traffic is routed through persistent connections that do not support load balance. As a result, the Grafana dashboard does not display the Transaction Per Second (TPS) for these pods.

Associate pods to nodes

After you have configured a cluster, you can associate pods to the nodes through labels. This association enables the pods to get deployed on the appropriate node based on the key-value pair.

Labels are required for the pods to identify the nodes where they must get deployed and to run the services. For example, when you configure the protocol-layer label with the required key-value pair, the pods get deployed on the nodes that match the key-value pair.

To associate pods to the nodes through the labels, use this configuration:

SUMMARY STEPS

1. Enter the label configuration for each node layer.

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	Enter the label configuration for each node layer.	`config`
		`  label`
		`  cdl-layer`
		`  key key_value`
		`  value value`
		`  oam-layer`
		`  key key_value`
		`  value value`
		`  protocol-layer`
		`  key key_value`
		`  value value`
		`  service-layer`
		`  key key_value`
		`  value value`
		`  end`
		NOTES:
		• If you opt not to configure the labels, then CPC assumes the labels with the default key-value pair.
		• **cdl-layer** — configures the key-value pair parameters for the CDL (Cisco Data Layer).
		• **oam-layer** — configures the key-value pair parameters for the OAM (Operations, Administration, and Maintenance) layer.
		• **protocol-layer** — configures the key-value pair parameters for the protocol layer.
		• **service-layer** — configures the key-value pair parameters for the service layer.

Sample configuration

```
pcf# show running-config label
label protocol-layer key smi.cisco.com/node-type-1
label protocol-layer value protocol
label service-layer key smi.cisco.com/node-type-2
label service-layer value service
label cdl-layer key smi.cisco.com/node-type-3
label cdl-layer value session
label oam-layer key smi.cisco.com/node-type
label oam-layer value oam
pcf#
```

View pod details and status

If a service requires additional pods, CPC creates and deploys the pods. You can view the list of pods participating in your deployment through the CPC Ops-Center.

You can run **kubectl** commands from the control-plane node to manage Kubernetes resources.

SUMMARY STEPS

1. To view a summary of pod details filtered by namespace, run one of the commands.
   - View pods across all namespaces filtered by CPC namespace:

     ```
     kubectl get pods -A | grep cpc-namespace
     ```

   - View pods with wide output including node assignment:

     ```
     kubectl get pods -A -owide | grep cpc-namespace
     ```

   - View all pods that are not in a Running state:

     ```
     kubectl get pods -A | grep -iv running
     ```

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	To view a summary of pod details filtered by namespace, run one of the commands.
	• View pods across all namespaces filtered by CPC namespace:
	`kubectl get pods -A | grep cpc-namespace`
	• View pods with wide output including node assignment:
	`kubectl get pods -A -owide | grep cpc-namespace`
	• View all pods that are not in a Running state:	

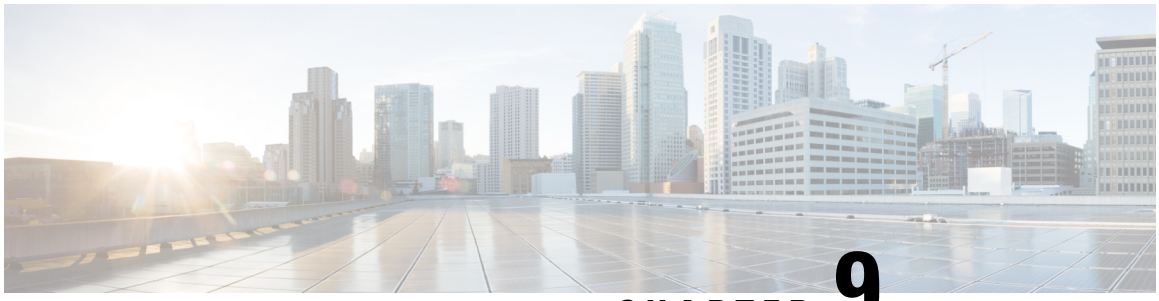
	Command or Action	Purpose
	`kubectl get pods -A | grep -iv running`	

Pod states

Understanding a pod's state lets you determine current health and prevent potential risks.

Table 14: Pod states

State	Description
Running	The pod is healthy and deployed on a node. It contains one or more containers.
Pending	The application is in the process of creating the container images for the pod.
Succeeded	All containers in the pod have successfully terminated. These pods cannot be restarted.
Failed	One or more containers in the pod have failed the termination process. The failure occurred because a container either exited with a non-zero status or the system terminated the container.
Unknown	The state of the pod could not be determined. This state typically occurs because the node where the pod resides was not reachable.



CHAPTER 9

Advanced Service Configurations

Overview

In CPC, reference data is considered information that is needed to operate the policy engine, but not used for evaluating policies. For example, in the **Reference Data** tab in CPC, are the forms used to define systems, clusters, and instances, and to set times and dates used for tariff switching. The policy engine needs to refer to this data only to process policies correctly. However, the data does not define the policy itself.

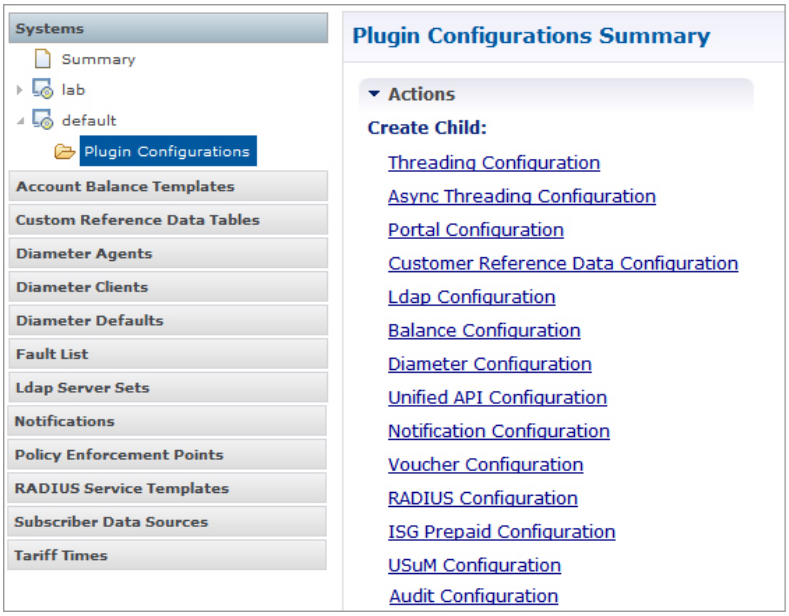
CPC provides core plug-ins for customizing and optimizing your installation.

- Configurations set at the system level are system-wide except as noted in the bullet items below.
- Configurations set at the cluster level apply to that cluster and the instances in it. A value set here overrides the same value set at the system level.
- Configurations set at the instance level apply to the instance only and override the same value set at the cluster or system level.

Select the **Create Child** action in a **Plug-in Configuration** node in the **Systems** tree to define them. You can change any of the variables from the default, or choose not to use a plug-in, as necessary.

When you create a system from the example, the following configuration stubs appear at the cluster and instance level:

Figure 21: Create Child Action



You are notified when a new policy is applied that overrides the existing configuration.

The notification is displayed as a warning icon above the configuration heading. When you hover over the warning icon, it displays the notification message as a tooltip. When there is an error and warning in the plugin configuration, then the error is overridden by a warning message.

A warning message is displayed under the following conditions:

- At the System level, if the selected plugin configuration is overridden by cluster or Instance plugin configuration.
- At the Cluster level, if the selected plugin configuration overrides the same plugin configuration at the system level or is overridden by the same plugin configuration at an Instance level.
- At the Instance level, if the selected plugin configuration overrides the same plugin configuration at system or cluster level.

Threading Configuration

A threading configuration utility is provided for advanced users.

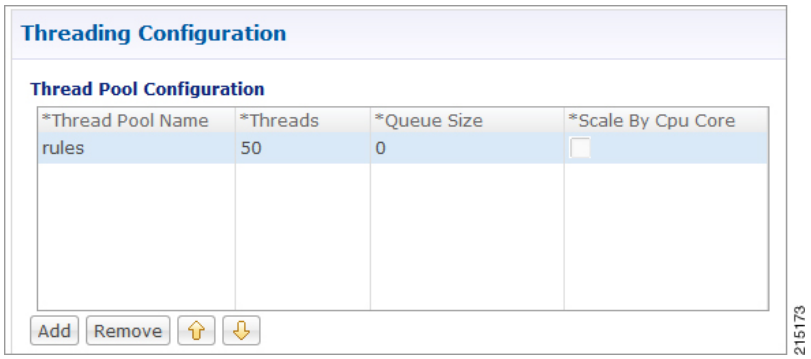
Click **Threading Configuration** in the right pane to add the threading configuration to the system. This is a performance tuning parameter and can be changed in case of a performance issue according to the call model and hardware. For further information, contact your Cisco Account representative.

The Threading Plug-in is for Mobility. The only value to set is **rules**. It controls the total number of threads in the Policy Engine (QNS) that are executing at any given time. The default value is 50.

It is recommended not to configure the value below 50. It can be set higher to help increase performance in certain situations where the queue full issue or performance issue is being observed. The value also depends on call model, hardware type.

A configuration example is shown below:

Figure 22: Thread Pool Configuration



The following parameters can be configured under Threading Configuration:

Table 15: Threading Configuration Parameters

Parameter	Description
Thread Pool Name	Name of the Cisco thread pool i.e., rules.
Threads	Specify the threads to set in the thread pool. You can set rules thread ranging from 50 to 100 depending on the call flow (based on number of lookup operations). • rules = 50; Queue Size = 0; Scale By Cpu Core = unchecked • rules = 100; Queue Size = 0 (If TPS is > 2000 per Policy Server (QNS) depending on call model used; Scale By Cpu core = unchecked The threads are driven based upon average response time of the message. The response time is call model dependent.
Queue Size	Specify the size of the queue before the threads are rejected. If value is greater than 50, performance may degrade because it holds the number of tasks in queue waiting for threads to be executed when TPS is high. If the value is lower than 50, the requests start dropping when all worker threads are busy in executing actions. The queue belongs to each Policy Server (QNS) process, and it holds incoming messages from Policy Directors (LB), but also internal events/messages (for example, an internal time change that triggers a policy evaluation). This is a performance tuning parameter and can be changed in case of a performance issue according to the call model and hardware. Default value is 0. **Note** In most of the setups, keep the queue size value default.

Parameter	Description
Scale By Cpu Core	Select this check box to enable the processor cores to scale the maximum number of threads. By default, this check box is unchecked.

Diameter Configuration

The Diameter Configuration section allows for the configuration of the diameter plug-in. It is recommended to configure the diameter plug-in at system level.

In order to define a Diameter Configuration at system level, perform these steps:

1. Login into Policy Builder.
2. Select **Reference Data** tab.
3. From the left pane, select **Systems**.
4. Select and expand your *system name*.
5. Select **Plugin Configurations**.
6. Select **Diameter Configuration**.

At Cluster Level

In order to define a Diameter Configuration at cluster level perform these steps:

1. Log in into Policy Builder.
2. Select **Reference Data** tab.
3. From the left pane, select **Systems**.
4. Select and expand your *system name*.
5. Select and expand your *cluster name*. If no cluster has been created, create one by selecting the **Cluster** action.
6. Select **Plugin Configurations**.
7. Select **Diameter Configuration**.

These parameters can be configured under Diameter Configuration.

Table 16: Diameter Configuration Parameters

Parameter	Description
Default Gx Stale Session Timer Minutes	This timer is armed every time a message is received or sent for any given Gx session. When the timer expires (or more precisely within the next minute after the timer expires) a Gx RAR having the Re-Auth-Request-Type AVP set to AUTHORIZE_ONLY (0) message is triggered for that Gx session. If a Gx RAA is received having Result-Code AVP value set to DIAMETER_UNKNOWN_SESSION_ID (5002) or DIAMETER_UNABLE_TO_COMPLY (5012) the Gx session is deemed as stale and removed from the PCRF internal database. On any activity over Gx interface (RAR/CCR) the timer is reset. Default value is 180 minutes. **Note** • This timer unit is minute. • Stale Gx session removal triggers the PCRF session termination procedure for any other diameter sessions that were bound to the Gx session.
Use V9 Event Trigger Mapping	This option allows for a different set of list of valid values and their interpretation to be used for the Event-Trigger enumerated AVP in order to accommodate the change that occurred in the Gx specification between 3GPP TS 29.212 v9.5 (and prior) and 3GPP TS 29.212 v9.7 (and following including v10 v11 and v12). Default value is checked. **Note** • The Event-Trigger AVP list of valid values and their interpretation defined in 3GPP TS 29.212 v9.6 is not supported. • Use V9 Event Trigger Mapping checked uses 3GPP TS 29.212 v9.5 as a reference while 3GPP TS 29.212 v110.10 is used as a reference when not checked.
Rel8 Usage Monitoring Supported	This option allows for the Gx usage monitoring feature to be supported even when the PCEF advertises support for Rel8 feature under Supported-Features AVP in Gx CCR-i. Default value is checked.
Rel15 Ext Bw Nr Supported	When checked, it enables the support for 3GPP Rel-15 Extended BW-NR feature. PCRF sends response to AF/PCEF with Rel-15 Extended BW-NR feature bit set when the feature is enabled and AF/PCEF has also set the bit in the request message. PCRF sends extended QoS AVPs only if the configuration is enabled. When unchecked, it disabled the support for 3GPP Rel-15 Extended BW-NR feature. Default value is unchecked (disabled).

Parameter	Description
Stale Session Configuration	When a new row is added in "Stale Session Configuration" table the default value for the GX_TGPP SD_V11 and SY_V11 Stale session timer is 180 minutes. **Note** The maximum value allowed for the **Stale Session Timer** parameter is 35000 minutes. If GX_TGPP Stale Session Timer value is not configured in this table, then the value is selected from the retained/old variable "Default Gx Stale Session Timer Minutes". If there are multiple values configured against any interface then the lowest among all would be considered as the stale session timer.

DRMP Prioritization	When enabled allows you to configure different message processing priorities based on the DRMP value received in the incoming request.
	• Default Inbound Priority - The default inbound priority value.
	• Inbound DRMP Prioritization
	  • DRMP - DRMP AVP value in the incoming request.
	  • Priority - Priority value assigned to the incoming message. Based on this value the message processing is prioritized. Higher Priority messages are be processed first compared to lower priority messages.

Note If Gx stale session timer is set for both "Default Gx Stale Session timer Minutes" and "Stale Session Configuration" then the value configured Under "Stale Session Configuration" would take the precedence.

Table 17: Use V9 Event Trigger Mapping Valid Values

Interpretation - Use V9 Event Trigger Mapping is checked	Value	Interpretation - Use V9 Event Trigger Mapping is not checked
SGSN_CHANGE	0	SGSN_CHANGE
QOS_CHANGE	1	QOS_CHANGE
RAT_CHANGE	2	RAT_CHANGE
TFT_CHANGE	3	TFT_CHANGE
PLMN_CHANGE	4	PLMN_CHANGE
LOSS_OF_BEARER	5	LOSS_OF_BEARER
RECOVERY_OF_BEARER	6	RECOVERY_OF_BEARER
IP_CAN_CHANGE	7	IP_CAN_CHANGE

Interpretation - Use V9 Event Trigger Mapping is checked	Value	Interpretation - Use V9 Event Trigger Mapping is not checked
QOS_CHANGE_EXCEEDING _AUTHORIZATION	11	QOS_CHANGE_EXCEEDING _AUTHORIZATION
RAI_CHANGE	12	RAI_CHANGE
USER_LOCATION_CHANGE	13	USER_LOCATION_CHANGE
NO_EVENT_TRIGGERS	14	NO_EVENT_TRIGGERS
OUT_OF_CREDIT	15	OUT_OF_CREDIT
REALLOCATION_OF_CREDIT	16	REALLOCATION_OF_CREDIT
REVALIDATION_TIMEOUT	17	REVALIDATION_TIMEOUT
UE_IP_ADDRESS_ALLOCATE	18	UE_IP_ADDRESS_ALLOCATE
UE_IP_ADDRESS_RELEASE	19	UE_IP_ADDRESS_RELEASE
DEFAULT_EPS_BEARER _QOS_CHANGE	20	DEFAULT_EPS_BEARER _QOS_CHANGE
AN_GW_CHANGE	21	AN_GW_CHANGE
SUCCESSFUL_RESOURCE _ALLOCATION	22	SUCCESSFUL_RESOURCE _ALLOCATION
RESOURCE_MODIFICATION _REQUEST	23	RESOURCE_MODIFICATION _REQUEST
PGW_TRACE_CONTROL	24	PGW_TRACE_CONTROL
UE_TIME_ZONE_CHANGE	25	UE_TIME_ZONE_CHANGE
USAGE_REPORT	26	TAI_CHANGE
TAI_CHANGE	27	ECGI_CHANGE
ECGI_CHANGE	28	CHARGING_CORRELATION _EXCHANGE
CHARGING_CORRELATION _EXCHANGE	29	APN_AMBR_MODIFICATION _FAILURE
USER_CSG_INFORMATION _CHANGE	30	USER_CSG_INFORMATION _CHANGE

Interpretation - Use V9 Event Trigger Mapping is checked	Value	Interpretation - Use V9 Event Trigger Mapping is not checked
DEFAULT_EPS_BEARER _QOS_MODIFICATION_FAILURE	31	NA
NA	33	USAGE_REPORT
	34	DEFAULT_EPS_BEARER _QOS_MODIFICATION_FAILURE
	35	USER_CSG_HYBRID _SUBSCRIBED_INFORMATION _CHANGE
	36	USER_CSG_HYBRID _UNSUBSCRIBED _INFORMATION_CHANGE
	37	ROUTING_RULE_CHANGE
	39	APPLICATION_START
	40	APPLICATION_STOP
	42	CS_TO_PS_HANDOVER
	43	UE_LOCAL_IP_ ADDRESS_CHANGE
	44	HENB_LOCAL_IP_ ADDRESS_CHANGE
	45	ACCESS_NETWORK_ INFO_REPORT

Next Hop routing

While selecting the peer that is used to deliver the request (with or without using the Next Hop Routes table) load balancing across the peers having the same rating is done. Load balancing starts from the peers having highest rating and covers all the peers in a round robin manner. If none is UP load balancing is tried with the peers having the second highest rating and again covers all the peers in a round robin manner and so on.

Note Next Hop Routes table is used only for PCRF initiated requests. The response messages for any incoming request is always delivered on the same connection where the request was received or not delivered at all. This is in order to avoid asymmetric routes.

The DRA should explicitly advertise support for a Diameter application other than Relay. The Relay application having Application Identifier 0xffffffff is not supported.

These parameters can be configured under Next Hop routing table:

Table 18: Next Hop Routing Parameters

Parameter	Description
Next Hop Realm	DRA realm name as received in Origin-Realm AVP in CER or CEA message. **Note** All the next hop realms (Next Hop Realm) should match the Origin-Realm AVP value in the incoming CER/CEA message.
Next Hop Hosts	DRA hosts name list as received in Origin-Host AVP in CER or CEA message. **Note** All the next hop host names (Next Hop Hosts) should match the Origin-Host AVP value value in the incoming CER/CEA message.
Application Id	Diameter application id advertised as being supported by the DRA. It contains information that identifies the particular service that the service session belongs to.
Destination Realms Pattern	Actual destination realm name pattern as received in Origin-Realm AVP in AAR message. The pattern needs to follow the standard Java regular expression syntax described here.
Destination Host Pattern	Actual destination host name pattern as received in Origin-Host AVP in AAR message. The pattern needs to follow standard Java pattern conventions. The pattern needs to follow the standard Java regular expression syntax described here.

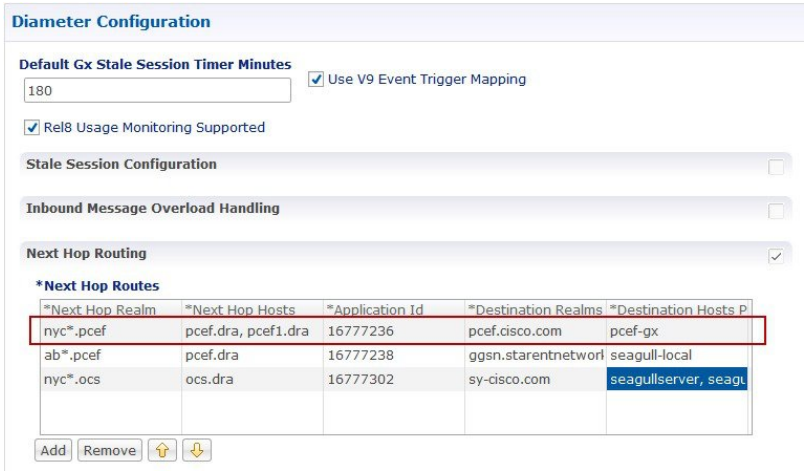
While populating the Next Hop Routes table, we recommend that you create only one entry for each Next Hop Realm value - Application Id value pair while all the DRA host names are provided as a list under Next Hop Hosts field. This is not a requirement though.

Note The order in which the DRA hosts are provisioned in the Next Hop Hosts field for any given next hop route is not relevant. The DRA host having the highest rating (priority) value is used. In case multiple hosts have the same rating one is randomly selected. Outbound realm rating of next hop is not considered except for SY_PRIME.

CPC supports grouping of realms and application identifiers using wildcarding and assigns it to a group of next hop peers. CPC routes outgoing messages by selecting the peer with highest priority.

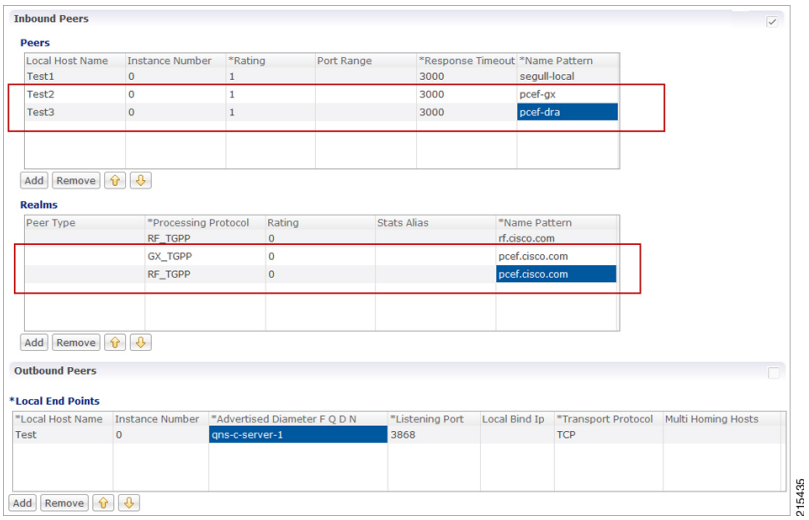
An example configuration for Grouping and Wildcarding in the Next Hop Routing table is shown below:

Figure 23: Grouping and Wildcarding in the Next Hop Routing Table



Destination Realm and Destination Hosts are used to map with the Peer configuration as defined in the Diameter Stack. The figure given below shows the mapping of the message containing the Realm from a peer to a protocol or interface.

Figure 24: Rating



Diameter Stack Configuration

You can enable the Diameter endpoint to dynamically create pods on a designated node or host. This feature might be a requirement when you want to ensure that the nodes are meeting specific security and regulatory parameters, or the node is closer to the data-center in terms of geographical proximity. The node affinity determines the node where CPC creates the Diameter endpoint pods, which are based on the affinity towards a node or group of nodes. Node affinity is a set of rules that allows you to define the custom labels on nodes and specify the label selectors within the pods. Based on these rules, the scheduler determines the location where the pod can be placed.

Note If you do not specify a node, then the Kubernetes scheduler determines the node where the Diameter endpoint creates a pod.

CPC supports both IPv4 and IPv6 connectivity on its external interfaces/endpoints (inbound and outbound).

Configuring the Node for the Diameter Endpoint Pod

This section describes how to specify the node or host where the Diameter endpoint must spawn the pod.

Note Configuration changes to the diameter endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.

To specify the node where you want Diameter endpoint to spawn the pod, use this configuration:

```
config
   diameter group diameter_group_name
   mode server server_name
   stack stack_name
     application application_name
     bind-ip ipv4 host_address
     bind-ipv6 ipv6 host_address
     bind-port port_number
     fqdn fqdn_address
     realm realm_address
     node-host node_host_address
     end
```

Note

- **diameter group** diameter_group_name —Specify the Diameter group name.
- **mode server** server_name —Specify the server name that operates as the mode server.
- **stack** stack_name —Specify the stack name.
- **application** application_name —Specify the application name.
- **bind-ip** host_address —Specify the host address IPv4 to bind the stack.
- **bind-ipv6** host_address —Specify the host address IPv6 to bind the stack.
- **bind-port** port_number —Specify the port number to bind the stack.
- **fqdn** fqdn_address —Specify the FQDN address.
- **realm** realm_address —Specify the realm address.
- **node-host** node_host_address —Specify the host IP address of the node.

Sample Configuration

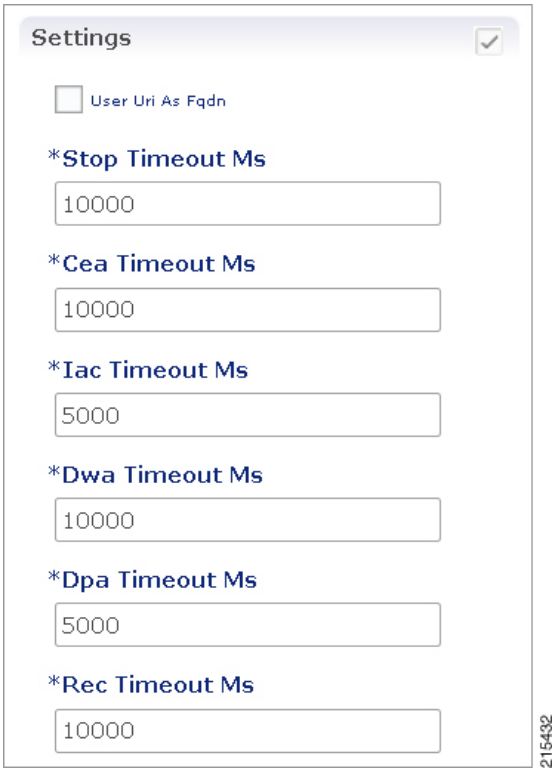
This is a sample configuration of the node configuration.

```
mode server
  stack cicdsite
  application rx
  bind-ip 192.0.2.18
  realm cisco.com
  node-host for-node-2a-worker39e1587354h
  exit
```

Settings

You can provision different timers that are available at the diameter stack level.

Figure 25: Settings



These parameters can be configured under Settings:

Table 19: Settings Parameters

Parameter	Description
User Uri As Fqdn	Sets the Origin-Host AVP value in CER/CEA to the user URI value instead of FQDn value. Default value is not set.
Stop Timeout Ms	Sets the timeout duration for a stack to wait till all the resources stop. The delay is in milliseconds. Default value is 10000.

Parameter	Description
Cea Timeout Ms	Sets the CER or CEA exchange timeout duration in case of no response. The delay is in milliseconds. Default value is 10000.
Iac Timeout Ms	Sets the timeout duration for a waiting stack before retrying the communication with a peer that has stopped answering DWR messages. The delay is in milliseconds. Default value is 5000.
Dwa Timeout Ms	Sets the DWR or DWA exchange timeout duration in case of no response. The delay is in milliseconds. Default value is 10000.
Dpa Timeout Ms	Sets the DPR or DPA exchange timeout duration in case of no response. The delay is in milliseconds. Default value is 5000.
Rec Timeout Ms	Sets the timeout duration for reconnection procedure. The delay is in milliseconds. Default value is 10000.

Dynamic ARP Functionality for PC and PV

CPC supports the dynamic ARP feature to send the same Priority-Level value in the dedicated bearers as that of the default bearer.

The dynamic ARP functionality is extended to Preemption Capability (PC) and Preemption Vulnerability (PV).

The PC parameter defines whether a bearer with a lower priority level can be dropped to free up the required resources.

The PV parameter defines whether a bearer is applicable for such dropping by a preemption capable bearer with a higher priority value.

To support this functionality for Rx interface, add two new columns, Rx_Dynamic_Vulnerability and Rx_Dynamic_Capability to the Rx_QoS_Table and for N5 interface, add two new columns, N5_Dynamic_Vulnerability and N5_Dynamic_Capability to the N5_QoS_Table.

How Dynamic ARP Works

This section describes how this feature works.

For a WPS user, the default bearer ARP value includes a Priority-Level value with PC set to enabled and PV set to disabled.

In case, when a non-WPS user calls a WPS user in Rx interface, the dynamic ARP attribute in the Rx_QoS_Table initiates the CPC to set the Priority-Level value in the dedicated bearer rules to match that of

the default bearer value. But the PVI/PCI values sent in the dedicated bearer rules use the enforced values from the Rx_QoS_Table (typically PVI enabled, PCI disabled).

In case, when a non-WPS user calls a WPS user in N5 interface, the dynamic ARP attribute in the N5_QoS_Table initiates the CPC to set the Priority-Level value in the dedicated bearer rules to match that of the default bearer value. But the PVI/PCI values sent in the dedicated bearer rules use the enforced values from the N5_QoS_Table (typically PVI enabled, PCI disabled).

For WPS user, if dynamic ARP attribute for PVI and PCI is set to "D", then the PVI and PCI values will be mirrored from the default bearer instead using the configured Rx QoS Table values in Rx interface and configured N5 QoS Table values in N5 interface.

Configure CRD table and RxSTG configuration AVP

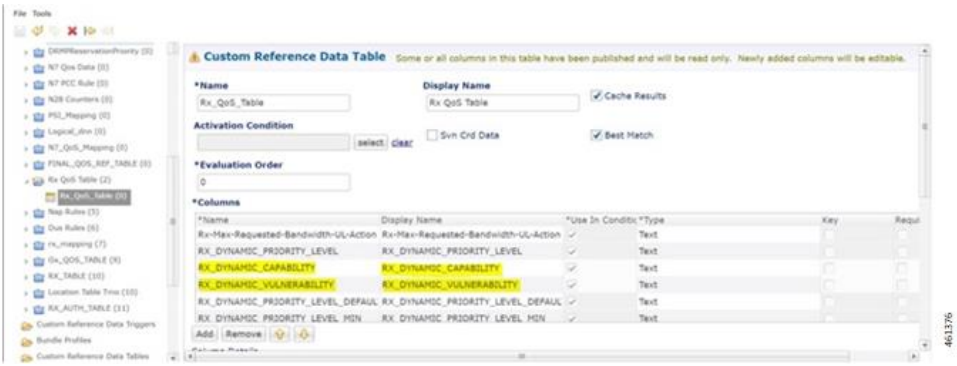
Configuring CRD table and RxSTG configuration AVP involves these steps:

Adding Rx Dynamic capability and Rx Dynamic vulnerability

To add Rx_Dynamic_Capability and Rx_Dynamic_Vulnerability columns to the Rx_QoS CRD table, use these steps:

1. Log in to Policy Builder.
2. Click the **Reference Data** tab, and from the left pane click **Custom Reference Data Tables** to view the options.
3. On the left pane, expand the **Search Table Groups** folder.
4. Expand the **Rx_QoS_Table** sub folder of **Search Table Groups** and click the.**Rx_QoS_Table**
5. Go to the ***Columns** field and click the **Add**.
6. Add the column **Name** and **Display Name** as **RX_DYNAMIC_CAPABILITY** and **RX_DYNAMIC_VULNERABILITY**.

Figure 26: Adding Rx_Dynamic_Capability and Rx_Dynamic_Vulnerability



Configuring RxSTGConfiguration AVP

This section describes the parameters that can be configured for RxSTGConfiguration.

The RxSTGConfiguration service configuration supports the following output AVPs that allow the dynamic value expression.

Before setting the service parameters, ensure that you create a use case template and add a service for this configuration.

The following table describes the RxSTGConfiguration service parameter.

Table 20: RxSTGConfiguration ParameterD

Parameters	Description
Dynamic-QoS-ARP-Pre-Emption-Capability	If the value is configured as "D" then the feature is enabled for PC. If the value is configured with any other value except "D" or is empty then the feature is disabled for PC.
Dynamic-QoS-ARP-Pre-Emption-Vulnerability	If the value is configured as "D" then the feature is enabled for PV. If the value is configured with any other value except "D" or is empty then the feature is disabled for PV.

Configuring CRD Table and N5STGConfiguration AVP

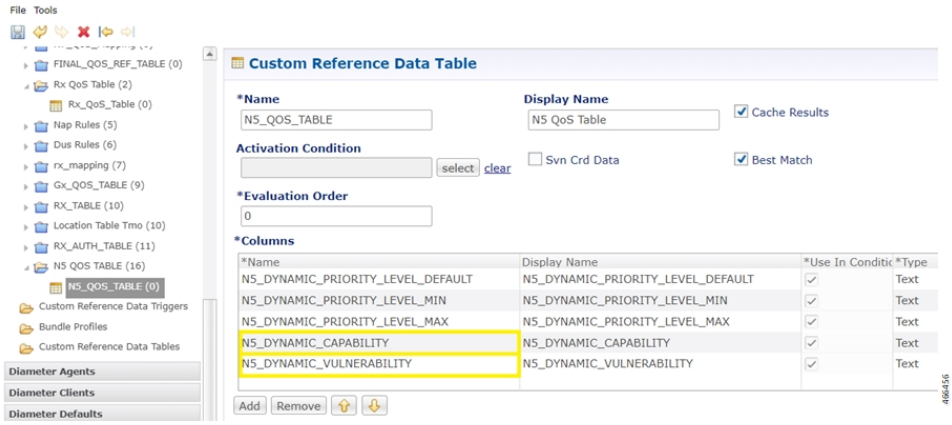
Configuring CRD table and N5STGConfiguration AVP involves these steps:

Adding N5_Dynamic_Capability and N5_Dynamic_Vulnerability

To add N5_Dynamic_Capability and N5_Dynamic_Vulnerability columns to the N5_QoS CRD table, use these steps:

1. Log in to Policy Builder.
2. Click the **Reference Data** tab, and from the left pane click **Custom Reference Data Tables** to view the options.
3. On the left pane, expand the **Search Table Groups** folder.
4. Expand the **N5_QoS_Table** sub folder of **Search Table Groups** and click the **N5_QoS_Table**.
5. Go to the ***Columns** field and click the **Add**.
6. Add the column **Name** and **Display Name** as **N5_DYNAMIC_CAPABILITY** and **N5_DYNAMIC_VULNERABILITY**.

Figure 27: Adding N5_Dynamic_Capability and N5_Dynamic_Vulnerability



Configuring N5STGConfiguration AVP

This section describes the parameters that can be configured for N5STGConfiguration.

The N5STGConfiguration service configuration supports the following output AVPs that allow the dynamic value expression.

Before setting the service parameters, ensure that you create a use case template and add a service for this configuration.

The following table describes the N5STGConfiguration service parameter.

Table 21: N5STGConfiguration ParameterD

Parameters	Description
Dynamic-QoS-ARP-Pre-Emption-Capability	If the value is configured as "D" then the feature is enabled for PC. If the value is configured with any other value except "D" or is empty then the feature is disabled for PC.
Dynamic-QoS-ARP-Pre-Emption-Vulnerability	If the value is configured as "D" then the feature is enabled for PV. If the value is configured with any other value except "D" or is empty then the feature is disabled for PV.

OAM Support

This section describes operations, administration, and maintenance support for this feature

Bulk Statistics Support

These statistics are supported for the dynamic ARP functionality for PC and PV feature.

Note These values apply to all the statistics:

- Unit - Int64
- Type - Counter
- Nodes - Service

- qos_rule_pc_total - Indicates the number of N5/N7/Rx rule installs (per qci/Media Type) provisioned with dynamic QoS PCI.

  The following labels are defined for this metric:

  - Interface
    - N5
    - N7
    - Rx
  - type
    - default_qos_pc
    - dynamic_qos_pc
  - identifier
    - qci
    - media-type
  - arp_pc

- qos_rule_pv_total - Indicates the number of N5/N7/Rx rule installs (per qci/Media Type) provisioned with dynamic QoS PVI.

  These labels are defined for this metric:

  - Interface
    - N5
    - N7
    - Rx
  - type
    - default_qos_pv
    - dynamic_qos_pv
  - identifier

- qci
- media-type

- arp_pv

Modified Stats

Table 22: Modified Stats

Old Stats	New Stats	Description

qos_rule_total	qos_rule_pl_total	Indicates the number of N5/N7/Rx rule installs (per qci/Media Type) provisioned with dynamic QoS PL.
		These labels are defined for this metric:
		• Interface
		  • N5
		  • N7
		  • Rx
		• type
		  • default_qos_pl
		  • dynamic_qos_pl
		• identifier
		  • qci
		  • media-type
		• arp_pl

Feature Description

CPC supports Diameter application KPI's and Alerts support in parity with PCRF application.

How It Works

This section describes how this feature works.

Statistics

node[x].messages.e2e __[realm_] Gx_CCR-I_2001. qns_stat.success

Description: Success message Policy Director count for return code 2001

node[x].messages.e2e __[realm_] Gx_CCR-I_2001. qns_stat.total _time_in_ms

Description: Total milliseconds Policy Director of successful messages with return code matching 2001

node[x].messages.e2e __[realm_] Gx_CCR-I_3xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_CCR-I_4xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_CCR-I_5xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 5XXX

node1.counters. [realm_] Gx_CCR-I.qns_count

Description: Count of messages Policy Server (qns) successful sent to the policy engine

node[x].messages. e2e__ [realm_] Gx_CCR-U_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Gx_CCR-U_2001. qns_stat.total_ time_in_ms

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e__ [realm_] Gx_CCR-U_3xxx. qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_CCR-U_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_CCR-U_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node1.counters. [realm_] Gx_CCR-U. qns_count

Description: Count of messages Policy Server (qns) successful sent to the policy engine

node[x].messages. e2e__ [realm_] Gx_CCR-U_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Gx_CCR-U_2001. qns_stat. total_time_in_ms

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e__ [realm_] Gx_CCR-U_3xxx. qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_CCR-U_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_CCR-U_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node1.counters. [realm_] Gx_CCR-U. qns_count

Description: Count of messages successful sent to the policy engine

node1.counters. [realm_] Gx_CCR-T. qns_count

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Gx_CCR-T_2001. qns_stat.success

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages.e2e_<domain>_[realm_]Gx_CCR-T_3xxx.qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_CCR-T_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_CCR-T_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node1.counters. [realm_] Gx_CCR-T.qns_count

Description: Count of messages successful sent to the policy engine

node1.counters. [realm_] Gx_RAR-T. qns_count

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Gx_RAR-T_2001. qns_stat.success

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages.e2e_<domain>_[realm_]Gx_RAR-T_3xxx.qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_RAR-T_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_RAR-T_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node[x].messages. e2e__ [realm_] Gx_RAR_timeout. qns_stat.success

Description: Success timeout Policy Director count for RAR message

node1.counters. [realm_] Gx_RAA.qns_count

Description: Count of all messages sent to the policy engine

node1.messages. in_q_Gx_RAA. qns_stat.error

Description: Count of messages failed to be sent to the policy engine

node1.messages. in_q_Gx_RAA. qns_stat.success

Description: Count of messages successful sent to the policy engine

node1.counters. [realm_] Gx_RAR.qns_count

Description: Count of messages successful sent to the Policy Director (LB)

node[x].messages. e2e__ [realm_] Rx_AAR_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Rx_AAR_2001. qns_stat.total_ time_in_ms

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e__ [realm_] Rx_AAR_3xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Rx_AAR_4xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Rx_AAR_5xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 5XXX

node1.counters. [realm_] Rx_RAA.qns_count

Description: Count of messages successful sent to the Policy Director (LB)

node1.counters. [realm_] Rx_AAR_drop. qns_count

Description: Count of messages dropped due to exceeding SLA

node1.counters. [realm_] Rx_AAA_2001. qns_count

Description: Count of AAA messages with result-code = 2001 sent successfully to the Policy Director (LB)

node[x].messages. e2e__ [realm_] Rx_ASR_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Rx_ASR_2001. qns_stat.total_ time_in_ms

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e__ [realm_] Rx_ASR_3xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Rx_ASR_5xxx. qns_stat.success

Description: Success count of Policy Director messages with return code matching 5XXX

node1.counters. [realm_] Rx_ASA_bypass. qns_count

Description: Count of message that do not require processing by the policy engine

node1.counters. [realm_]Rx_ASA. qns_count

Description: Count of messages successful sent to the policy engine

node1.counters. [realm_] Rx_ASA_drop. qns_count

Description: Count of messages dropped due to exceeding SLA

node[x].messages. e2e__ [realm_] Rx_RAR_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Rx_RAR_2001. qns_stat.total_ time_in_ms

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e_<domain>_[realm_] Gx_RAR-T_3xxx. qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Gx_RAR-T_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Gx_RAR-T_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node1.counters. [realm_] Rx_RAA_bypass. qns_count

Description: Count of messagethat do not require processing by the policy engine

node1.counters. [realm_] Rx_RAA.qns_count

Description: Count of message successful sent to the policy engine

node1.counters. [realm_] Rx_RAA_drop. qns_count

Description: Count of messages dropped due to exceeding SLA

node[x].messages. e2e__ [realm_] Rx_STR_2001. qns_stat.success

Description: Success message count for return code 2001

node[x].messages. e2e__ [realm_] Rx_STR_2001. qns_stat.total_time_in_m

Description: Total milliseconds of successful messages with return code matching 2001

node[x].messages. e2e__ [realm_] Rx_STR_3xxx. qns_stat.success

Description: Success count of messages with return code matching 3XXX

node[x].messages. e2e__ [realm_] Rx_STR_4xxx. qns_stat.success

Description: Success count of messages with return code matching 4XXX

node[x].messages. e2e__ [realm_] Rx_STR_5xxx. qns_stat.success

Description: Success count of messages with return code matching 5XXX

node1.counters. [realm_] Rx_STR.qns_count

Description: "Count of messages successful sent to the policy engine"

node1.counters. [realm_] Rx_STR_drop. qns_count

Description: Count of messages dropped due to exceeding SLA

node1.messages. in_q_Rx_STR. qns_stat.success

Description: "Count of messages successful sent to the policy engine"

node1.messages. in_q_Rx_STR. qns_stat. total_time_in_ms

Description: Total milliseconds of messages successfully sent to the policy engine

node1.messages. diameter_Rx_STR. qns_stat.success

Description: Success message count

node1.messages. diameter_Rx_STR. qns_stat. total_time_in_ms

Description: Total milliseconds of successful messages

node1.counters. [realm_] Rx_STA_2001. qns_count

Description: Count of STA messages with result-code = 2001 sent successfully to the Policy Director (LB)

Alarms

RxAAR

Description: "This alert is fired when the percentage of Success Rx AAR responses send is lesser threshold."

Formula:

sum(increase(diameter_responses_total{command_code=\”AAA\”,response_status=~\”2001\”}[5m])) / sum(diameter_responses_total(outgoing_request_total{command_code=\”AAA\”}[5m])) < 0.90”

RxSTA

Description: "This alert is fired when the percentage of Success Rx STA responses send is lesser threshold."

Formula:

sum(increase(diameter_responses_total{command_code=\”STA\”,response_status=~\”2001\”}[5m])) /sum(diameter_responses_total(outgoing_request_total{command_code=\”STA\”}[5m])) < 0.90

RxRAR

Description: "This alert is fired when the percentage of Success Rx RAR responses Received is lesser threshold."

Formula:

sum(increase(diameter_responses_total{command_code=\”RAA\”, response_status=~\”2001\”}[5m]))/sum(diameter_responses_total(outgoing_request_total{command_code=\”RAA\”}[5m])) < 0.90

RxASR

Description: "This alert is fired when the percentage of Success Rx ASR responses send is lesser threshold."

Formula:

sum(increase(diameter_responses_total{command_code=\”ASA\”,response_status=~\”2001\”}[5m])) /sum(diameter_responses_total(outgoing_request_total{command_code=\”ASA\”}[5m])) < 0.90

pod-down

Description: CDL EP Pod Down

Formula:

up{pod=~'cdl-ep.*'} == 0

pod-down

Description: CDL Pod Slot Change

Formula:

up{pod=\"cdl-slot-session-c1-m1-0\"} == 0

pod-down

Description: Diameter EP Change

Formula:

up{pod=~'diameter-ep.*'} == 0

pod-down

Description: EP Mapping Change

Formula:

up{pod=~'etcd-pcf.*'} == 0

pod-down

Description: Grafana Dashboard Change

Formula:

up{pod=~'grafana-dashboard.*'} == 0

pod-down

Description: Kafka Changed

Formula:

up{pod=~'kafka.*'} == 0

pod-down

Description: LDAP Pod Changed

Formula:

up{pod=~'ldap-pcf.*'} == 0

pod-down

Description: CPC Engine Changed

Formula:

up{pod=~pcf-engine-pcf.*'} == 0

pod-down

Description: CPC Rest EP Change

Formula:

up{pod=~'pcf-rest-ep.*'} == 0

LDAP Query

Description: "This alert is fired when the success percentage of ldap query request is lesser threshold."

Formula:

sum(increase(message_total{type=~\".*_ldap_query\", status=\"success\"}[5m]))
/sum(increase(message_total{type=~\".*_ldap_query\"}[5m])) < 0.90

LDAP Modify

Description: "This alert is fired when the success percentage of ldap modify request is lesser threshold."

Formula:

sum(increase(message_total{component=\"ldap-ep\", type=~\".*_ldap_modify\",status=\"success\"}[5m]))
/ sum(increase(message_total{component=\"ldap-ep\",type=~\".*_ldap_modify\"}[5m])) < 0.90

PLF Request

Description: This alert is fired when the success percentage of PLF request is lesser threshold.

Formula:

sum(increase(message_total{type=~\"ldap_search-res_success\",status=\"success\"}[5m]))
/sum(increase(message_total{type=~\"ldap_search-res_.*\"}[5m])) <0.90

NAP Notification

Description: This alert is fired when the success percentage of NAP request is lesser threshold.

Formula:

sum(increase(message_total{type=~\"ldap_change-res_success\", status=\"success\"}[5m]))
/sum(increase(message_total{type=~\"ldap_change-res_.*\"}[5m])) <0.90

node-disk-running-full

Description: test

Formula:

node_filesystem_usage > 0.0001

vm-down

Description: VM Down

Formula:

up{pod=~\"node-expo.*\"} == 0

mem-util-high

Description: High Memory Usage

Formula:

avg(node_memory_MemAvailable_bytes /node_memory_MemTotal_bytes * 100) by (hostname) < 20

disk-util-high

Description: High Disk Usage

Formula:

avg (node_filesystem_avail_bytes{mountpoint =\"/\"} /node_filesystem_size_bytes{mountpoint =\"/\"} *100) by (hostname) <20

cpu-util-idle

Description: High CPU Usage

Formula:

avg(rate(node_cpu_seconds_total{mode='idle'}[1m])) by (hostname) *100 < 50

Implementation of SPR MongoDB in CPC

The CPC supports efficient subscriber authorization for Gx and N7 interfaces through integration with SPR. This integration enables the CPC to provide internal access to the SPR and the Unified API Endpoint, facilitating seamless policy management and updates.

The key aspects include:

- **Subscriber Session Policy Review:**

  Upon receiving notifications of any changes in subscriber profiles stored in the SPR database, the CPC automatically reviews the associated subscriber session policies.

- **Policy Update Notifications:**

  If a subscriber policy update is necessary following a profile change, the CPC initiates the appropriate actions by sending a Gx-Re-Authorization Request (Gx-RAR) or an N7 Notify message.

- **Seamless Integration with SMF:**

  In the event of a policy change, the CPC uses the N7 Notify message to communicate with the Session Management Function (SMF), ensuring that all sessions are updated in accordance with the latest subscriber policies.

The SPR MongoDB Deployment in CPC for Subscriber Data Handling feature involves deploying MongoDB replica sets within the CPC Kubernetes cluster namespace. These replica sets are crucial for storing policy and charging control information of subscribers, which are provisioned through the CPC Unified API endpoint. The SPR deployed with Geo Redundancy - Multisite MongoDB Cluster with site local Unified API and API Router support.

In the case of Embedded Subscriber Profile Repository (SPR), CPC supports subscriber authorization for Gx and N7. The CPC provides internal access to both SPR and the Unified API Endpoint. The SPR databases get configured in an existing administration database replica set or used as a separate MongoDB replica set.

CPC must review the subscriber session policies if any subscriber profile changes get notified in the SPR DB. If there is any post profile update in the subscriber policy, the CPC sends a Gx-RAR or N7 Notify message. When the CPC receives the notification, it locates the subscriber profile from the SPR DB, reviews the policy, and initiates an N7 Notify message to SMF if there is any policy change.

Configure SPR Mongo Replica Sets in CPC

This procedure allows to configure SPR mongo replica sets in Ops-Center and in policy builder (PB) page.

Procedure

Step 1 Login to the Global Configuration mode.

```
config // Enter config mode
```

Step 2 Enter the given CLI command for CDL configuration to retrieve the N7 session which needs to be re-evaluated while there is a change in subscriber profile:

```
cdl datastore session
find-by-nuk-prefixes USuMMissingCredKey
exit

cdl datastore session
find-by-nuk-prefixes USuMSubscriberIdKey
exit
```

Step 3 Enter the CLI command for CPC engine configuration.

```
engine <ENGINE_NAME>
properties usum.missingcredential.keyname value USuMMissingCredKey
exit
```

Note

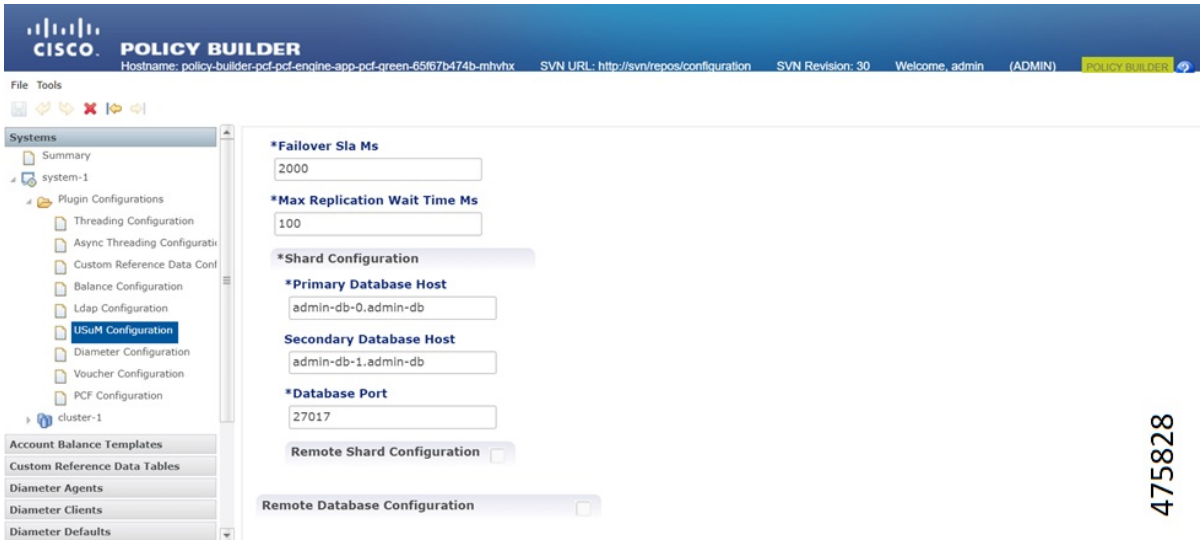
- Add these configurations when all the pods are in shutdown state.
- Here is the sample configuration of mongo replica set in Ops-Center:

```
db scdb replica-name mongo-spr1
 port 27018
 interface db1.mdb.2564
 resource memory limit 112000
 replica-set-label key smi.cisco.com/node-type-5
 replica-set-label value db-spr
 member-configuration member sprdb-rs1-arbiter
  host 10.160.158.195
  arbiter true
  site remote
 exit
 member-configuration member sprdb-rs1-s1-m1
  host 10.160.9.196
  arbiter false
  priority 166
  site remote
 exit
 member-configuration member sprdb-rs1-s1-m2
  host 10.160.9.197
  arbiter false
  priority 155
  site remote
 exit
 member-configuration member sprdb-rs1-s2-m1
  host 10.160.49.196
  arbiter false
  priority 66
  site local
 exit
 member-configuration member sprdb-rs1-s2-m2
  host 10.160.49.197
  arbiter false
  priority 55
  site local
```

```
 exit
exit
db scdb replica-name mongo-spr2
 port 27019
 interface db1.mdb.2564
 resource memory limit 112000
 replica-set-label key smi.cisco.com/node-type-5
 replica-set-label value db-spr
 member-configuration member sprdb-rs2-arbiter
  host 10.160.158.195
  arbiter true
  site remote
 exit
 member-configuration member sprdb-rs2-s1-m1
  host 10.160.9.198
  arbiter false
  priority 166
  site remote
 exit
 member-configuration member sprdb-rs2-s1-m2
  host 10.160.9.199
  arbiter false
  priority 155
  site remote
 exit
 member-configuration member sprdb-rs2-s2-m1
  host 10.160.49.198
  arbiter false
  priority 66
  site local
 exit
 member-configuration member sprdb-rs2-s2-m2
  host 10.160.49.199
  arbiter false
  priority 55
  site local
 exit
exit
```

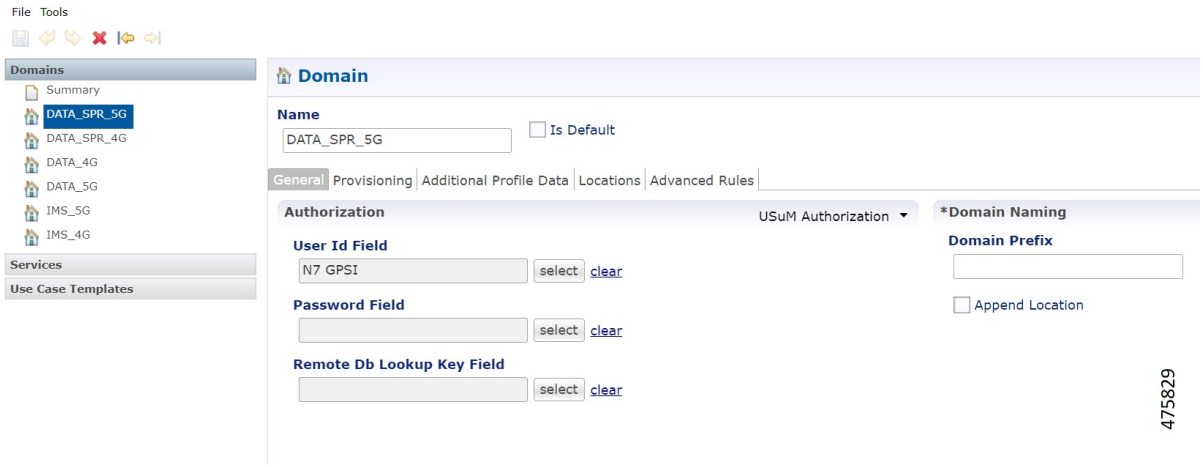
Step 4 Login to policy builder page and select **Plugin Configuration** option. Click the **USuM Configuration** option.

Figure 28: USuM Configuration



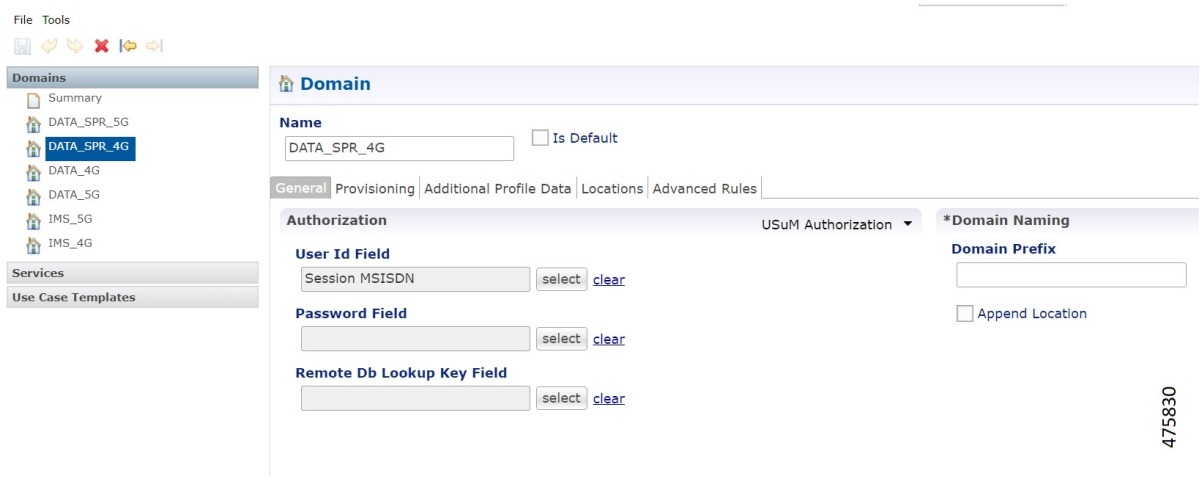
Step 5 Use the N7 GPSI or SUPI value as a lookup key for N7 configuration in SPR.

Figure 29: SPR N7 Configuration



Step 6 In the **Domain** tab, select the **USuM Authorization** option from drop down. You can use the Gx IMSI or MSISDN as a lookup key for Gx configuration.

Figure 30: SPR Gx Configuration



API Router Functionality for Subscriber Data Storage

The API Router functionality allows the CPC to store and manage subscriber data across multiple SPR replica sets. This is particularly useful for managing large volumes of subscriber data and ensuring high availability and redundancy. The feature also supports multi-credential management, enabling the system to handle subscribers with multiple identifiers.

Multi-Credential Management

The IMSI or MSISDN number of a subscriber is the primary networkId of a subscriber. CPC supports creation or modification action for subscribers using the IMSI and MSISDN numbers. The MSISDN and IMSI numbers can have the same or different ending starting from 0 through 4 or from 5 through 9.

Use these API queries to create or edit subscriber data:

- CreateSubscriberRequest: Create a subscriber with both IMSI and MSISDN credentials
- GetSubscriberRequest: Query a subscriber by either IMSI or MSISDN.

Note To skip the error code when the subscriber is not available, set the **uapi.get.sub.skip.fail.response** CLI to true.

- DeleteSubscriberRequest: Delete a subscriber by either IMSI or MSISDN.
- ChangeCredentialUsernameRequest: Change the credential for a subscriber by either IMSI or MSISDN.

Here is an example for subscriber creation:

```
<CreateSubscriberRequest>
            <subscriber>
                    <credential>
                            <networkId>5678</networkId>
```

```
                <type>IMSI</type>
            </credential>
            <credential>
                <networkId>1234</networkId>
                <type>primary</type>
            </credential>
            <credential>
                <networkId>bob</networkId>
            </credential>
             ...
        </subscriber>
    </CreateSubscriberRequest>
```

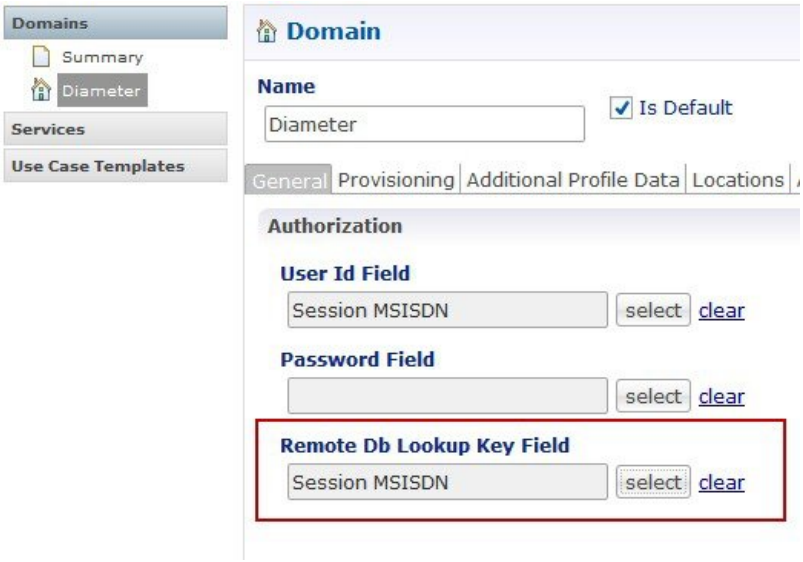
Set Network ID in Domain Configuration

Use the MSISDN or IMSI number as a network ID of a subscriber to support multiple SPR.

Procedure

Step 1 Click the **Services** option on right side of Policy Builder page.

Step 2 In the **Domain** tab on right pane, select the **Remote Db Lookup Key Field** as **Session MSISDN** or **Session IMSI** option.



Configure API router to route subscriber data

Configure the API router to route the subscriber data based on the last digit of the networkId. Here, the SPR1 stores the subscriber information for networkId that ends with 0 through 4 and SPR2 stores the subscriber information for networkId that ends with 5 through 9.

Before you begin

To install the API router feature, configure the installation feature parameters in Ops Center. Login to the configuration mode and set the CLI configuration for engine:

```
engine <ENGINE_NAME> install-features policy-builder com.broadhop.client.feature.apirouter
engine <ENGINE_NAME> install-features policy-server com.broadhop.apirouter.service.feature
```

Procedure

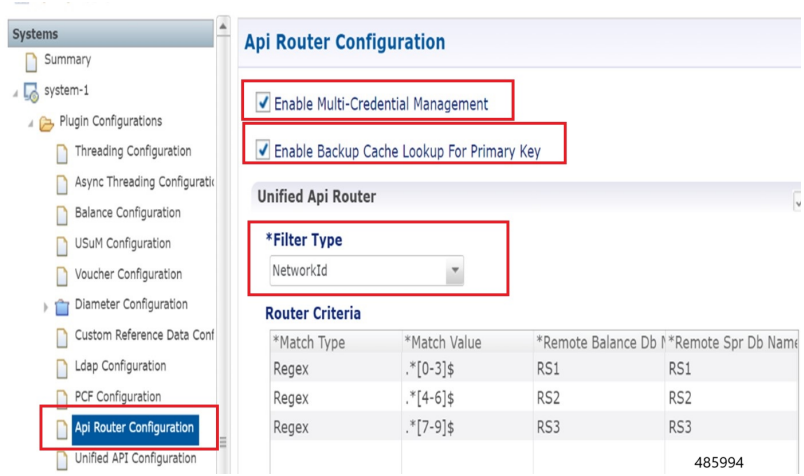
Step 1 In the left pane of Policy Builder page, click **Plugin Configurations** option.

Step 2 Click the **API Router Configuration** tab.

Step 3 Select the **Enable Multi-Credential Management** check box.

Step 4 Select the **Enable Backup Cache Lookup For Primary Key** check box.

Step 5 Set the **Filter Type** as **NetworkId** to configure mutiple SPR. The **Router Criteria** displays remote SPR database values from **USuM Configuration** and Remote Balance Database from **Balance Configuration**.



Configure Remote Database with Balance Configuration

You can configure the remote balance database in Policy Builder page using Balance configuration tab.

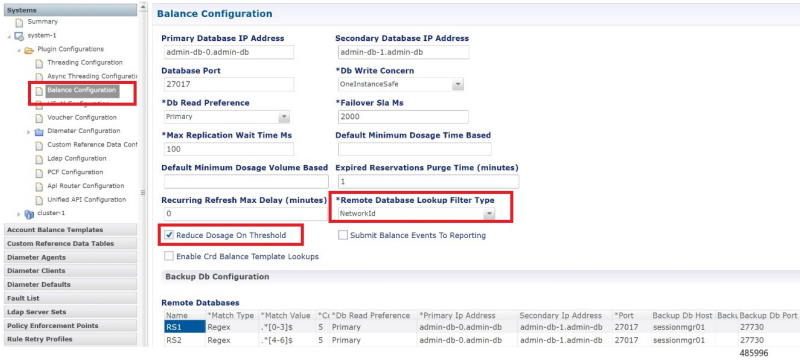
Procedure

Step 1 In the left pane of Policy Builder page, click **Plugin Configurations** option.

Step 2 Select **Balance Configuration** tab.

Step 3 Enter the mandatory values.

Step 4 By default, the **Remote Database Lookup Filter Type** is set as **Networkid** and the **Reduce Dosage on Threshold** check box is selected.



Configure Remote Database with USuM Configuration

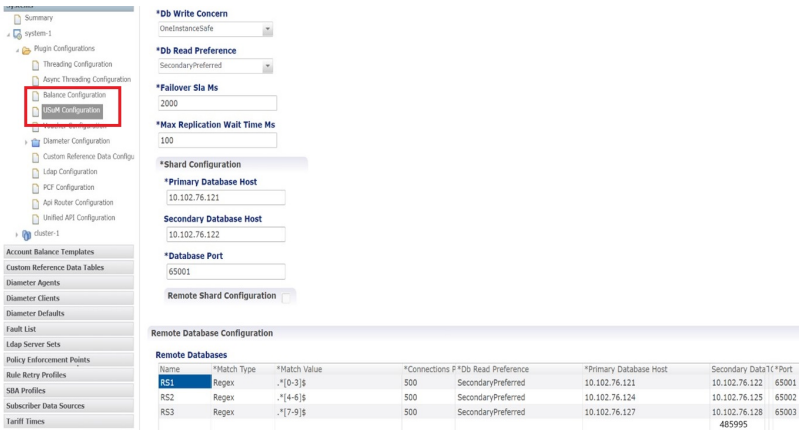
You can configure the remote SPR database in Policy Builder page using UsuM configuration tab.

Procedure

Step 1 In the left pane of Policy Builder page, click **Plugin Configurations** option.

Step 2 Select **USuM Configuration** tab.

Step 3 Enter the mandatory Database values.



Dynamic QoS Uplift based on Subscriber QoS Schedule

The Dynamic QoS Uplift feature enables the adjustment of the QCI value for a subscriber session dynamically, based on predefined schedules. This manages the network resources and ensures optimal performance during peak times or for specific applications.

In addition to QCI uplift, CPC supports dynamic uplift of Access Point Name (APN) Aggregate Maximum Bit Rate (AMBR) for subscriber sessions using the same QoS schedule mechanism. The APN AMBR uplift is provisioned through the Subscriber Profile Repository (SPR) and applied based on the configured APN,

validity time window, and priority. When the QoS schedule becomes active, CPC applies the provisioned uplink and downlink AMBR values to the matching subscriber sessions. At the end of the configured time window, CPC automatically reverts the AMBR values to those that were effective prior to the uplift. This enhancement ensures deterministic, time-bound bandwidth management and consistent QoS enforcement across Gx and N7 interfaces.

When multiple QoS schedules are provisioned for a subscriber, CPC evaluates the priority attribute associated with each schedule to determine the effective QoS profile. If priorities are configured, the QoS schedule with the highest priority takes precedence. If multiple schedules have the same priority, the schedules are rejected and the currently active QoS remains unchanged. If priority is not configured, CPC uses the start time to determine the applicable QoS schedule. This priority-based evaluation applies to both QCI and APN AMBR parameters.

QoS schedule management for a subscriber

Provision QoS schedule

Use these API queries to create or edit subscriber data and provision the QoS schedule for a subscriber:

- CreateSubscriberRequest
- UpdateSubscriberRequest

Here is an example to provision the QoS schedule:

```
<qosSchedule>
  <id>5</id>
  <qci>5</qci>
  <apn>APN_A</apn>
  <startDate>2024-08-27T11:50:00.000Z</startDate>
  <endDate>2024-08-27T11:50:59.000Z</endDate>
  <apnAmbrUl>45000</apnAmbrUl>
  <apnAmbrDl>45000</apnAmbrDl>
  <priority>1</priority>
</qosSchedule>
```

QoS schedule fields

The QoS schedule supports APN-level and DNN-level AMBR configuration with priority-based evaluation. A QoS schedule includes an identifier, a validity window, interface-specific QoS parameters, and an optional priority. The supported fields vary depending on the interface (Gx or N7).

Gx Interface

The following QoS schedule fields are supported for Gx-based subscriber sessions:

- `apnAmbrUl` (Integer): Specifies the uplink Access Point Name (APN) Aggregate Maximum Bit Rate (AMBR) applied while the QoS schedule is active.
- `apnAmbrDl` (Integer): Specifies the downlink Access Point Name (APN) Aggregate Maximum Bit Rate (AMBR) applied while the QoS schedule is active.

Here is an example to provision the QoS schedule:

```
<qosSchedule>
  <id>5</id>
  <gci>5</gci>
```

```
  <apn>APN_A</apn>
  <startDate>2025-10-17T11:50:00.000Z</startDate>
  <endDate>2025-10-17T11:51:00.000Z</endDate>
  <apnAmbrUl>45000</apnAmbrUl>
  <apnAmbrDl>45000</apnAmbrDl>
</qosSchedule>
```

Configuration for Gx

Figure 31: Gx-Based Subscriber QoS Schedule Configuration — USuMActiveQoSSchedule APN AMBR Fields

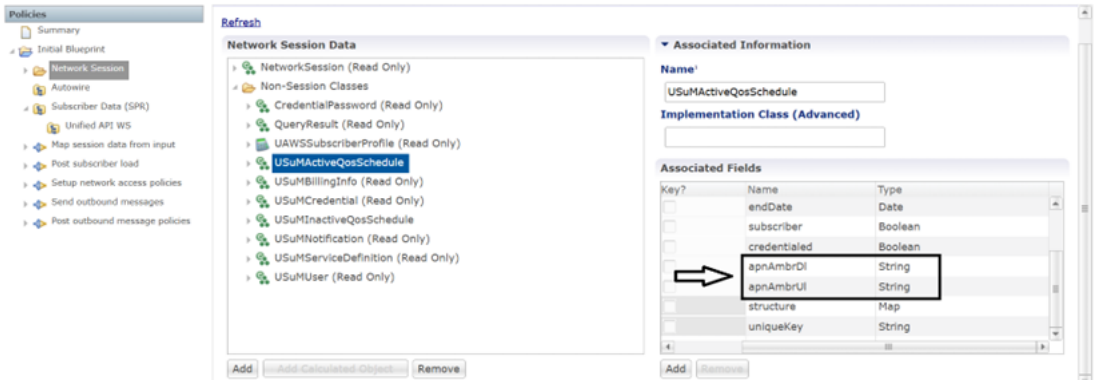
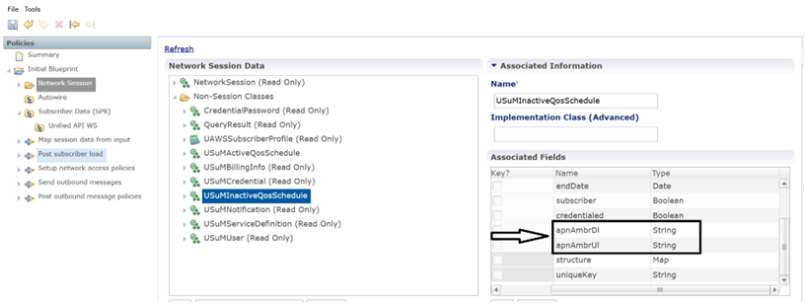


Figure 32: Gx-Based Subscriber QoS Schedule Configuration — USuMInactiveQoSSchedule APN AMBR Fields



N7 Interface

For N7-based sessions, CPC supports dynamic QoS scheduling using DNN-specific parameters. QoS schedules can include 5QI and aggregate DNN AMBR values with an optional priority.

When a QoS schedule becomes active, CPC applies the provisioned QoS parameters to matching subscriber sessions on the N7 interface based on the configured validity window and priority. Upon schedule expiry, CPC automatically restores the QoS values that were effective prior to the uplift.

The following QoS schedule fields are supported for N7-based subscriber sessions:

- `dnn` (String): Specifies the Data Network Name (DNN) to which the QoS schedule applies.
- `fiveQi` (Integer): Specifies the 5QI value applied while the QoS schedule is active.
- `dnnAmbrUl` (Integer): Specifies the uplink DNN Aggregate Maximum Bit Rate (AMBR) applied while the QoS schedule is active.
- `dnnAmbrDl` (Integer): Specifies the downlink DNN Aggregate Maximum Bit Rate (AMBR) applied while the QoS schedule is active.

- `priority`: Defines the precedence of the QoS schedule when multiple schedules exist for a subscriber.

Here is an example to provision the QoS schedule:

```
<qosSchedule>
  <id>1</id>
  <fiveQi>5</fiveQi>
  <dnn>DNN_A</dnn>
  <startDate>2025-10-17T11:50:00.000Z</startDate>
  <endDate>2025-10-17T11:51:00.000Z</endDate>
  <dnnAmbrUl>45000</dnnAmbrUl>
  <dnnAmbrDl>45000</dnnAmbrDl>
  <priority>1</priority>
</qosSchedule>
```

Configuration for N7

Figure 33: N7-Based Subscriber QoS Schedule Configuration — USuMActiveQoSSchedule dnn Fields

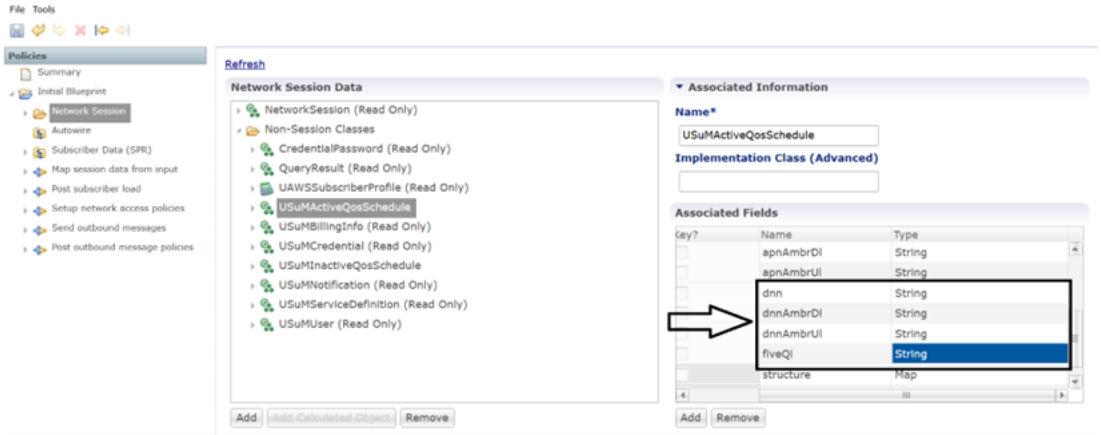
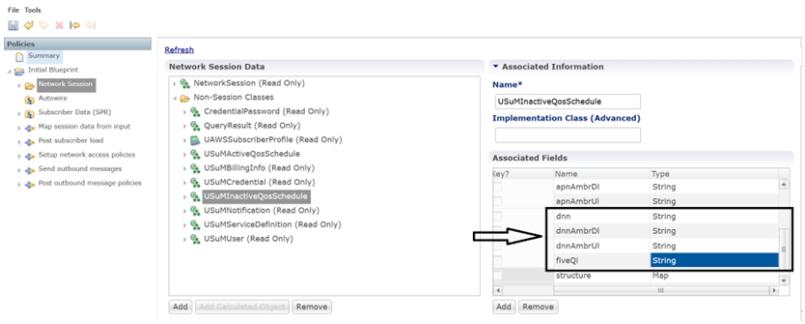


Figure 34: N7-Based Subscriber QoS Schedule Configuration — USuMInactiveQoSSchedule dnn Fields



Note

- The schedule includes the qosSchedule ID, QCI value, APN, start date, and end date.
- There is no tag to support all APNs. To enable for **ALL APN**, do not send **<apn>** tag in the QoS schedule request.
- In the start date and end date, the application supports hours and minutes fields without seconds. So, update the values as **00** seconds for start date and **59** seconds for end date fields.
- The `<apnAmbrUl>` and `<apnAmbrDl>` elements specify the uplink and downlink APN AMBR values to be applied when the QoS schedule is active.
- A lower numerical value indicates a lower priority. Priority comparison is applicable only when multiple QoS schedules are provisioned for a subscriber. When a single qosSchedule is provided and the priority attribute is not specified, CPC assigns a default priority value of 0
- If multiple QoS schedules have the same priority, CPC rejects those schedules and retains the existing QoS configuration.
- If priority is not specified and multiple schedules share the same start date, the schedules are rejected.
- If priority is not specified and start dates differ, CPC applies the QoS schedule based on the start date.
- If priority is specified for one QoS schedule and not specified for another, CPC assigns the lowest priority to the QoS schedule without an explicit priority value.

CPC sends a Re-Auth Request (RAR) to the Packet Gateway (PGW) to adjust the QCI value at the start and end times specified in the schedule. Multiple QoS schedules can be defined for different APNs for a single subscriber.

View QoS schedule information

Use the **GetSubscriberRequest** API to view the QoS Schedule information after provisioning the QoS schedule.

Delete expired QoS schedules

Send **UpdateSubscriberRequest** with empty values in the **qosSchedule** to delete all expired schedules.

Multiple session handling

CPC matches the sessions with the provisioned APN and send RARs to adjust the QCI value at the configured start and end times.

For subscribers with multiple active sessions using different APNs, CPC applies the APN AMBR uplift only to the sessions whose APN matches the value specified in the QoS schedule. Sessions established after the start time and before the end time of an active QoS schedule are immediately created with the provisioned AMBR values. At the end time, CPC reverts the AMBR values for the affected sessions to the values that were active before the QoS schedule was applied.

Feature Description

CPC needs to support configuration to accept inputs for next hop route for route rating, application Id, destination host, destination realm, next hop realm, next hop destination, and next hop destination rating.

When the diameter connection gets set up, the DRA displays the origin host and realm values, and all diameter application messages includes the actual host origin host and realm. CPC requires a method to determine the DRA connection that delivers the message to the desired host.

CPC sends the Diameter Request Messages to check the next hop route matches for the message's application Id, destination host, and destination realm. If there is any match for the next hop routes, the message is sent to the Diameter Peer mapped to the next hop destination and next hop realm.

For multiple next hop routes, there are multiple next-hop destinations with the same or different ratings. CPC needs to find the next hop realm and destination combo that carries higher ratings for the next next hop destination and deliver the message for a combination of the application Id, destination host, and destination realm..

How It Works

This section describes how Next Hop Routing Support feature works.

Feature Configuration

This section describes how to configure the Next Hop Routing Support.

To configure the Next Hop Routing Support, use this configuration:

```
config
  diameter
    next-hop-route* [application-id destination-realm destination-host
rating]
    application-id       -> /diameter/application/application-id
    destination-realm   string
    destination-host    string
    rating              int32
    next-hop-realm   string
    next-hop-host*  [next-hop-host-name]
    next-hop-host-name  string
        rating?         int32
        end
```

NOTES:

- **next-hop-route**—Specifies the next hop realm and next next-hop-host details for a combination of application-id, destination-host and destination-realm of the outgoing diameter request message.
- **application-id**—Specifies the Auth-Application-Id from diameter request message.
- **rating**—Specifies the rating of the route.
- **destination-realm**—Specifies the Destination-Realm from the diameter request message.

- **destination-host**—Specifies the Destination-Host from the diameter request message.
- **next-hop-realm**—Specifies the peer-realm configured in the diameter client stack at the CPC which need to be used for combination of application-id, destination-host and destination-realm in the diameter request message.
- **next-hop-host**—Specifies the details about the next hop host to be used with host name and rating:
  - **next-hop-host-name**—Specifies the peer-host configured in the diameter client stack at the CPC which need to be used for combination of application-id, destination-host and destination-realm in the diameter request message
  - **rating**—Specifies the rating of the next-hop-host-name. Higher rated destination host will be selected when multiple destination-hosts are configured and reachable.

Configure indirect SCP communication model D

Feature description

Table 23:

Feature Name	Release Information	Description
Configure indirect SCP communication model D	2026.03.0	The indirect Service Communication Proxy (SCP) Model D feature enables the Converged Policy (CPC) to discover and route through an SCP. This architecture centralizes policy decision points across 5G interfaces, which reduces configuration overhead by offloading discovery tasks to the SCP. This improves interoperability and simplifies session policy correlation in multi-vendor 5G core networks.

The indirect Service Communication Proxy (SCP) Model D enables the Converged Policy (CPC) and Policy Control Function (PCF) to discover and route traffic through an SCP rather than communicating directly with target Network Functions (NFs). This phase extends support to the Access and Mobility Management Function (AMF), Application Function (AF), Charging Function (CHF), and Unified Data Repository (UDR) across the N5, N15, N28, and N36 interfaces. This architecture centralizes policy decision points, which reduces configuration overhead by offloading discovery tasks to the SCP.

Configure the indirect SCP communication model

Before you begin

Before you configure this feature, ensure that you meet these requirements:

- Ensure the cluster runs version 1.24 or later.
- Confirm that all SCP endpoints are resolvable and reachable from the PCF REST-EP pods.
- Verify that existing Network Repository Function (NRF) discovery configurations for N5, N15, N28, and N36 are active to support `fallbackToDirect`.
- Ensure you have administrative privileges in the PCF Ops-Center.

Use this procedure to establish connectivity to the SCP middleware, enable the Model D feature, and set NF binding IDs.

Procedure

Step 1 Log in to the CPC Ops-Center CLI.

Step 2 Enter `config` mode and configure the CA certificate to establish trust with the SCP:

```
pcf-tls
 ca-certificates <ca-name>
  cert-data <ca-pem-data>
 exit
```

Step 3 Configure the server certificate for the CPC:

```
certificates <server-cert-name>
  cert-data <server-pem-data>
  private-key <server-private-key>
 exit
```

Step 4 `commit` the configuration.

Configure the SCP profile

Define the SCP profile to establish the connection path to the SCP middleware.

Procedure

Step 1 Enter `config` mode.

Step 2 Define the SCP profile using the `profile nf-client` structure:

```
profile nf-client nf-type scp
 scp-profile <scp-profile-name>
  locality <locality-name>
  service name type nscp-routing
   endpoint-profile <endpoint-profile-name>
    capacity <capacity-value>
    priority <priority-value>
    uri-scheme https
    version uri-version v1
    endpoint-name <endpoint-name>
```

```
     primary ip-address ipv4 <scp-ip> port <scp-port>
    exit
   exit
  exit
 exit
exit
```

Step 3 `commit` the changes.

Enable Model D and mapping interfaces

Enable the global Model D and assign the SCP profile to the specific interfaces.

Procedure

Step 1 Enter `config` mode.

Step 2 Enable the global Model D feature: `rest-endpoint model-d` enabled.

Step 3 Link the SCP profile defined in the previous procedure: `rest-endpoint model-d scp-profile <scp-profile-name> priority 1 locality <locality-name>`

Step 4 Enable the required interfaces for SCP routing:

```
rest-endpoint model-d interfaces n5-callbacks true

rest-endpoint model-d interfaces n15-callbacks true

rest-endpoint model-d interfaces n28-chf true

rest-endpoint model-d interfaces n36-udr true
```

Step 5 Configure the fallback behavior to ensure PCF reverts to direct NRF discovery if the SCP is unreachable:

```
rest-endpoint model-d fallbackToDirect true
rest-endpoint model-d scpMaxRetries 2
```

Step 6 `commit` the changes.

Configure NF Binding IDs

Configure the NF instance identity to ensure the SCP correctly identifies the CPC during routing.

Procedure

Step 1 Enter `config` mode.

Step 2 Configure the NF instance and set IDs:

```
service-registration profile nf-instance-id <instance-id>
service-registration profile nf-set set-id <set-id>
```

Step 3 `commit` the changes.

Verify the configuration

Use this procedure to validate that the system applies the Model D configuration.

Procedure

Step 1 Check the `pcf:model_d_enabled` gauge in the Prometheus dashboard to confirm the feature is active.

Step 2 Run `show running-config` command in the CPC Ops-Center to confirm the rest-endpoint model-d configuration is active.

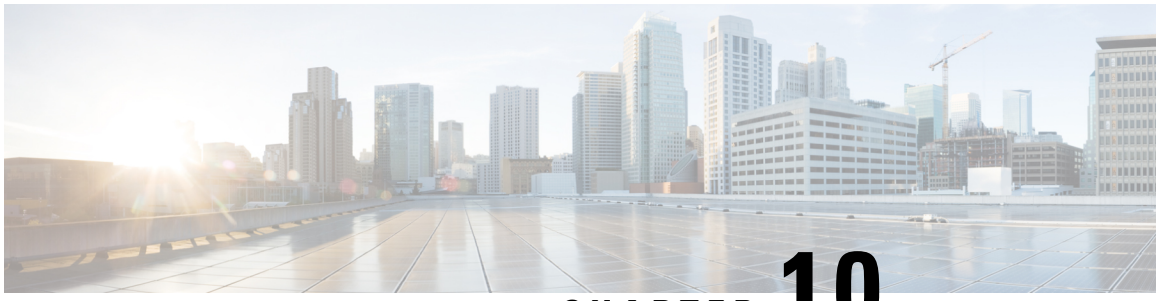
Step 3 Confirm that the CPC routes requests through the SCP by checking the `pcf:route_by_nftype_scp_total` metric.

```
rest-endpoint model-d enabled
rest-endpoint model-d scp-profile scp1.local priority 1 locality L1
rest-endpoint model-d scp-profile scp2.local priority 2 locality L1
rest-endpoint model-d fallbackToDirect true
rest-endpoint model-d scpMaxRetries 2
```

Interpret results and troubleshoot

Table 24: Diagnostic metrics and troubleshoot

Metric or issue	Interpretation or corrective action
`pcf:model_d_enabled` gauge	A value of `1` indicates the feature is active.
`pcf:route_by_nftype_scp_total`	Confirms successful request processing via the SCP relay.
`pcf:scp_retry_count`	Tracks the number of fallbacks or retries due to SCP unavailability.
SCP reachability failure	Verify SCP reachability if the `pcf:scp_retry_count` metric increases consistently.
Header parsing errors	Review CPC logs for header parsing errors if the SCP fails to route requests to the target NF.
Binding failure	Confirm that the NF instance ID and NF set ID are correctly configured in the CPC Ops-Center.
Header stripping	Ensure the SCP does not strip required headers by comparing incoming request headers against 3GPP specification requirements.



CHAPTER 10

Advanced Tuning Parameters

Converged Policy advanced tuning parameters

The CPC Ops Center allows you to configure the advanced tuning parameters for CPC. The tuning parameters primarily consist of the async-threading and http2-threading parameters. These parameters provide the flexibility of the tuning threads responsible for CPC's incoming and outgoing requests over HTTP.

Configure the advanced tuning parameter values only if you have a strong understanding of the CPC deployment.

Configuring the Async Threading Parameters

To configure the http2-threading parameters, use this configuration in the Policy Ops Center console:

```
config
  advance-tuning
  async-threading
    default-priority default_priority
    default-worker-threads default_worker_threads
    default-queue-size default_queue_size
    default-processing-threads default_processing_threads
    default-drop-oldest-when-full [ true | false ]
    threading-config service_name
    priority priority
    queue-size queue_size
    threads number_threads
    end
```

NOTES:

- **advance-tuning**—Enters the advance tuning configuration mode.
- **async-threading**—Enters the async threading configuration mode.
- **default-priority** *default_priority*—Specify the default priority level.

- **default-worker-threads** *default_worker_threads*—Specify the default number of worker threads.
- **default-queue-size** *default_queue_size*—Specify the default size of the queue.
- **default-processing-threads** *default_processing_threads*—Specify the default number of threads used for processing.
- **default-drop-oldest-when-full [true | false]** —Indicates if the oldest message in the queue should be removed when the queue is full.
- **threading-config** *service_name*—Specify the service name for which the threading configuration is enabled.
- **priority** *priority*—Specify the priority of the thread.
- **queue-size** *queue_size*—Specify the queue size.
- **threads** *number_threads*—Specify the number of threads to be processed.

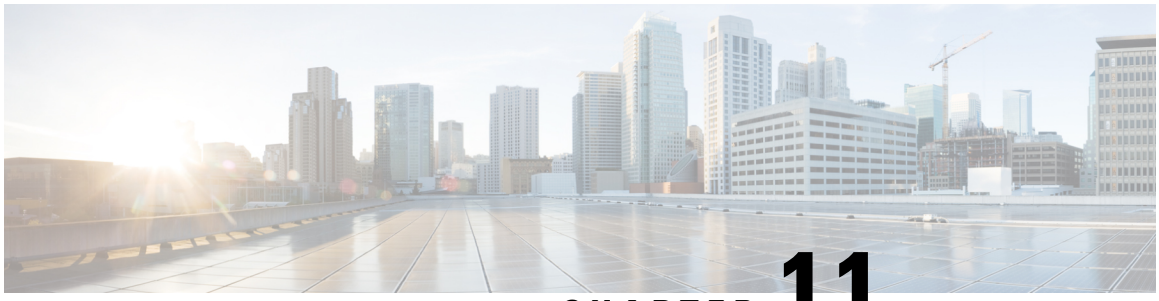
Configuring the HTTP2 Threading Parameters

To configure the http2-threading parameters, use this configuration in the Policy Ops Center console:

```
config
   http2-threading
   min-thread-pool-size min_thread_pool
   max-thread-pool-size max_thread_pool
     idle-thread-timeout-ms idle_thread_timeout
     max-queue-capacity max_queue_capacity
     disable-validation [ true | false ]
     end
```

NOTES:

- **http2-threading** http2_threading – Configures the parameters for inbound SBA requests that are received by CPC.
- **min-thread-pool-size** min_thread_pool – Specifies the minimum number of threads for processing the inbound SBA request. The accepted range contains integers. Default value is 5.
- **max-thread-pool-size** max_thread_pool – Specifies the maximum size of the thread pool.
- **idle-thread-timeout-ms** idle_thread_timeout – Specifies the time in milliseconds that the thread can remain idle. idle_thread_timeout must contain only integers. Default value is 60000.
- **disable-validation [true | false]** – Disables the validation of the request sent to CPC. **[true | false]** must contain the value as true or false. Default value is false.
- **max-queue-capacity** max_queue_capacity – Specifies the maximum number of requests that can wait in the queue for processing. max_queue_capacity must contain only integers. Default value is 5000.
- **max-thread-pool-size** max_thread_pool – Specifies the maximum number of threads that CPC can accommodate in the pool. max_thread_pool_size must contain only integers. Default value is 20.



CHAPTER 11

Diameter Endpoint

Feature Description

You can enable the Diameter endpoint to dynamically create pods on a designated node or host. This feature might be a requirement when you want to ensure that the nodes are meeting specific security and regulatory parameters, or the node is closer to the datacenter in terms of geographical proximity. The node affinity determines the node where CPC creates the Diameter endpoint pods, which are based on the affinity towards a node or group of nodes. Node affinity is a set of rules that allows you to define the custom labels on nodes and specify the label selectors within the pods. Based on these rules, the scheduler determines the location where the pod can be placed.

Note If you do not specify a node, then the Kubernetes scheduler determines the node where the Diameter endpoint creates a pod.

CPC supports both IPv4 and IPv6 connectivity on its external interfaces/endpoints (inbound and outbound).

Configuring the Node for the Diameter Endpoint Pod

This section describes how to specify the node or host where the Diameter endpoint must spawn the pod.

Note Configuration changes to the diameter endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.

To specify the node where you want Diameter endpoint to spawn the pod, use the following configuration:

```
config
   diameter group diameter_group_name
```

```
mode server server_name
stack stack_name
  application application_name
  bind-ip ipv4 host_address


  fqdn fqdn_address
  realm realm_address
  node-host node_host_address
  end
```

NOTES:

- **diameter group** *diameter_group_name*—Specify the Diameter group name.
- **mode server** *server_name*—Specify the server name that operates as the mode server.
- **stack** *stack_name*—Specify the stack name.
- **application** *application_name*—Specify the application name.
- **bind-ip** *host_address*—Specify the host address to bind the stack.
- **fqdn** *fqdn_address*—Specify the FQDN address.
- **realm** *realm_address*—Specify the realm address.
- **node-host** *node_host_address*—Specify the host IP address of the node.

Sample Configuration

The following is a sample configuration of the node configuration.

```
mode server
  stack cicdsite
  application rx
  bind-ip 192.0.2.18
  realm cisco.com
  node-host for-node-2a-worker39e1587354h
  exit
```

RX-AAR messages handling during site failover

Table 25: Feature History

Feature Name	Release Information	Description
Cross-Site Session Handling for Distributed CPC		This feature enables seamless cross-site session handling within a distributed Converged Policy (CPC) system. The feature enables CPC nodes to process Rx session-related messages for sessions created at other sites. When the system is configured to accept Diameter messages with unknown destination hostnames, session processing continues during site failovers, load balancing, or dynamic routing scenarios. **Command introduced:** **diameter properties**

Configuring diameter properties for cross-site session handling

This procedure describes how to enable CPC nodes to accept Diameter messages with unknown Destination-Host values, which is essential for cross-site session handling in a distributed CPC environment.

Before you begin

Ensure you meet these important prerequisites before you start with the Ops-Center configuration:

- Ensure you have administrative access to the system's Configuration Management Framework.
- Understand the impact of this configuration change on your distributed CPC system.

Follow these steps to configure Diameter properties for cross-site session handling.

Procedure

Step 1 Enter the Ops-Center to specify the Diameter properties.

Example:

```
PCF(config)# diameter properties
```

Step 2 Change the `jdiameter.accept.unknown_desthost` property value from false to True.

a) **false**: The Diameter protocol stack rejects messages where the Destination-Host does not match the hostname of the receiving CPC node, returning error code 3002 (DIAMETER_UNKNOWN_PEER).

b) **true**: The Diameter protocol stack accepts messages with unknown Destination-Host values and bypasses strict hostname validation. This allows the CPC to process the message locally or retrieve session state.

Example:

```
PCF(config)# diameter properties jdiameter.accept.unknown_desthost value true
Tue Sep  9  06:21:31.289 UTC+00:00
PCF(config-properties-jdiameter.accept.unknown_desthost)# commit
Tue Sep  9  06:21:34.615 UTC+00:00
```

After applying the changes, verify that the `jdiameter.accept.unknown_desthost` property is successfully set to `true` on the CPC nodes. You can verify the property by querying the configuration or by observing system behavior. Applying this property config redeploys the diameter pods.

Diameter message level overload handling

Feature description

Table 26: Feature description

Feature Name	Release Information	Description
Diameter message-level overload handling	2026.03.0	The Diameter message-level overload handling is an enhancement that provides ingress protection for the Converged Policy (CPC) Diameter endpoint. It mitigates traffic bursts that cause latency, request timeouts, and instability by enforcing traffic limits at the pod level before requests reach the engine. This ensures system reliability and protects resources from excessive traffic.

This enhancement enforces traffic limits at the `cps-diameter-ep` pod level for the Converged Policy (CPC) Diameter endpoint. It supports ingress traffic for Gx, Rx, and Sy interfaces. It mitigates traffic bursts that cause latency, request timeouts, and instability by enforcing traffic limits at the pod level before requests reach the `pcf-engine`.

- Global overload protection: Applies at the pod level, affecting all stacks within the `cps-diameter-ep` pod.
- Stack-specific overload protection: Applies to a specific stack within the `cps-diameter-ep` pod.

Applying overload protection globally and at the stack level for specific Diameter procedures protects system resources, prioritizes legitimate traffic, and minimizes the processing load on downstream components.

Configure global overload protection

Before you begin

Before you configure this feature, ensure that you meet these requirements:

- Ensure you have administrative access to the Converged Policy Ops-Center.
- Verify that the `cps-diameter-ep` pod is running and healthy.
- Confirm that you have the necessary permissions to modify the `advance-tuning` and `diameter` configuration hierarchies.

Use this procedure to configure global traffic limits and rejection actions for Diameter messages.

Procedure

Step 1 Log in to the Converged Policy Ops-Center and enter `config` mode.

Step 2 Set the global ingress request limit.

```
advance-tuning overload-control diameter global limits max-requests-per-sec <value>
```

Step 3 Define the global action for rejected requests.

```
advance-tuning overload-control diameter global action throttle-action <reject | drop>
```

Step 4 Specify the Diameter result code for rejected requests.

```
advance-tuning overload-control diameter global action threshold-reject-code <code_value>
```

Step 5 Configure message-specific threshold actions.

```
advance-tuning overload-control diameter global action threshold-action <command_name>
discard-action <reject | drop>
threshold-count <value>
exit
```

Step 6 `commit` the changes.

Configure stack-specific overload protection

Use this procedure to configure traffic limits and rules for specific Diameter stacks.

Procedure

Step 1 Enter `config` mode

Step 2 Define the stack-specific transactions per second (TPS) limit.

```
diameter group <group_name> stack <stack_name> overload-control limits max-requests-per-sec <value>
```

Step 3 Configure overload rules for the stack.

```
diameter group <group_name> stack <stack_name> overload-control rule <rule_name>
threshold-tps <value>
discard-action <reject | drop>
command-code [ <code_list> ]
request-type [ <type_list> ]
exit
```

Step 4 `commit` the changes.

Verify and troubleshoot configurations

Use this procedure to verify configurations and to troubleshoot potential issues.

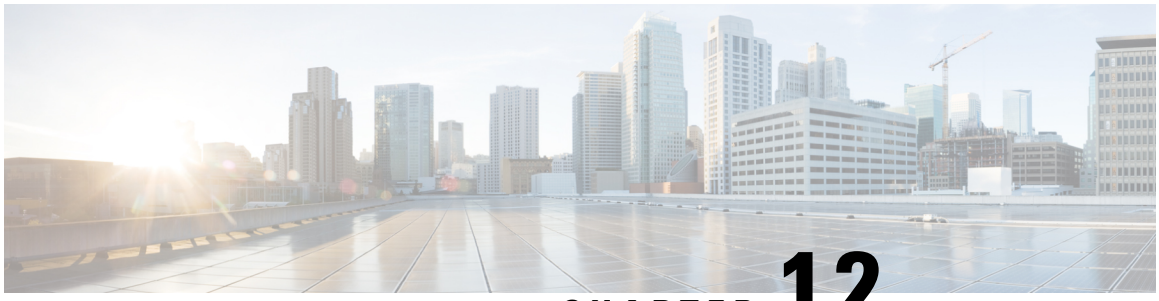
1. Run the `show running-config` command in the Converged Policy Ops-Center to ensure the output displays the intended global and stack-specific parameters.
2. Monitor the `cps-diameter-ep` Prometheus counters to confirm that the system tracks throttled, threshold-exceeded, and within-threshold requests

The `cps-diameter-ep` pod evaluates inbound Diameter requests against the defined global and stack-specific limits. The system rejects or drops requests exceeding these thresholds to prevent them from reaching the `pcf-engine`.

Troubleshoot

- Verify the syntax of the `advance-tuning` or `diameter` commands by using the `show configuration` command in the Converged Policy OpsCenter.
- Ensure that the `threshold-tps` or `max-requests-per-sec` values are set appropriately for the expected traffic volume.
- Check the Prometheus counters to determine if the traffic hits the global ingress limit before reaching stack-specific rules, as global limits take precedence.
- Verify the `DiameterServerStackListener` logs for runtime errors or rule-based throttling events:

  ```
  kubectl logs <cps-diameter-ep-pod-name>
  ```



CHAPTER 12

Application Based Alert

Feature Description

When the system detects an anomaly, it generates an alert notification. The system statistics are the cause for these alert notifications. You can set an expression to trigger an alert when the expression becomes true.

How it Works

This section describes how this feature works.

The Common Execution Environment (CEE) uses the Prometheus Alert Manager for alerting operations. The CEE YANG model, accessible through CLI or API, allows you to view the active alerts, silenced alerts, and alert history. During the application installation or upgradation, the system adds a set of preset alerting rules. Also, the applications can call the alert API directly to add or clear alerts. The Prometheus Alert Manager API (v2) is the standard API used.

The Prometheus Alerts Manager includes the following options:

- **DefiningAlert Rules**: This option defines the types of alerts that the Alert Manager should trigger. Use the Prometheus Query Language (PromQL) to define the alerts.
- **Defining Alert Routing**: This option defines the action the Alert Manager should take after receiving the alerts. At present, the SNMP Trapper is supported as the outbound alerting. Also, the CEE provides an Alert Logger for storing the generated alerts.

Configure alert rules

This section describes how to configure the alert rules.

To configure the alert rules, use this configuration:

```
config
   alerts rules group alert_group_name
   rule rule_name
      expression promql_expression
      duration duration
      severity severity_level
      type alert-type
      annotation annotation_name
      value annotation_value
      end
```

NOTES:

- **alerts rules**—Specify the Prometheus alerting rules.
- **group** *alert_group_name*—Specify the Prometheus alerting rule group. One alert group can have multiple lists of rules. *alert-group-name* is the name of the alert group. *alert_group_name* must be a string in the range of 0–64 characters.
- **rule** *rule_name*—Specify the alerting rule definition. *rule_name* is the name of the rule.
- **expression** *promql_expression*—Specify the PromQL alerting rule expression. *promql_expression* is the alert rule query expressed in PromQL syntax. The *promql_expression* must be a string in the range of 0–64 characters.
- **duration** *duration*—Specify the duration of a true condition before it is considered true. *duration* is the time interval before the alert is triggered.
- **severity** *severity_level*—Specify the severity of the alert. *severity-level* is the severity level of the alert. The severity levels are critical, major, minor, and warning.
- **type** *alert_type*—Specify the type of the alert. *alert_type* is the user-defined alert type. For example, Communications Alarm, Environmental Alarm, Equipment Alarm, Indeterminate Integrity Violation Alarm, Operational Violation Alarm, Physical Violation Alarm, Processing Error Alarm, Quality of Service Alarm, Security Service Alarm, Mechanism Violation Alarm, or Time Domain Violation Alarm.
- **annotation** *annotation_name*—Specify the annotation to attach to the alerts. *annotation_name* is the name of the annotation.
- **value** *annotation_value*—Specify the annotation value. *annotation_value* is the value of the annotation.

The following example configures an alert, which is triggered when the percentage of N7 responses is less than the specified threshold limit.

```
configure terminal
  alerts rules group PCFN7chk_incr
  interval-seconds 300
  rule PCFN7chk_incr
  expression "sum(increase(inbound_request_total{interface_name=\"N7\",
result_code=~\"2..\"}[3m])) / sum(increase(inbound_request_total{interface_name=\"N7\"}[3m]))<
 0.95"
    severity major
    type "N7 Communications Alarm"
    annotation summary
    value "This alert is fired when the percentage of N7 responses is less than threshold‟
    exit
  exit
exit
```

View alert logger

The alert logger stores the alerts that CPC generates by default. View these alerts using the this command:

show alert history [filtering]

Narrow down the result using these filtering options:

- **annotations**—Specify the annotations of the alert.
- **endsAt**—Specify the end time of the alert.
- **labels**—Specify the additional labels of the alert.
- **severity**—Specify the severity of the alert.
- **source**—Specify the source of the alert.
- **startsAt**—Specify the start time of the alert.
- **type**—Specify the type of the alert.

You can view the active and silenced alerts with the **show alerts active** and **show alerts active** commands.

```
show running-config alerts
  interval-seconds 300
  rule PCFN7chk_incr
  expression "sum(increase(inbound_request_total{interface_name=\"N7\",
result_code=~\"2..\"}[3m])) / sum(increase(inbound_request_total{interface_name=\"N7\"}[3m]))<
 0.95"
    severity major
    type "N7 Communications Alarm"
    annotation summary
    value "This alert is fired when the percentage of N7 responses is less than threshold‟
    exit
  exit
exit
```

This section provides sample outputs for active and historical alerts in the CPC system.

```
show alerts history
alerts active PCFN7chk_incr ac2a970ab621
state active
severity major
type "N7 Communications Alarm"
startsAt 2019-11-15T08:26:48.283Z
source System
annotations [ "summary:This alert is fired when the percentage of N7 responses is less than
 threshold.” ]
```

The following example displays the active alerts. The alerts remain active as long as the evaluated expression is true.

```
show alerts active
alerts active PCFN7chk_incr ac2a970ab621
state active
severity major
type "N7 Communications Alarm"
startsAt 2019-11-15T08:26:48.283Z
source System
```

```
annotations [ "summary:This alert is fired when the percentage of N7 responses is less than
 threshold.” ]
```

Sample alerts configuration

This section provides sample configurations that are defined in CPC.

Interface-Specific Alerts

N7 Interface Inbound

Use these commands to configure alerts related to an inbound N7 interface.

```
alerts rules group PCFSvcStatus
   interval-seconds300
   rulePCFN7Inbound
   expression sum(increase(inbound_request_total{interface_name=\”N7\”,
result_code=~\”2..\”}[5m])) /sum(increase(inbound_request_total{interface_name =\”N7\”}[5m]))
 <0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the percentage of Success N7 responses sent is lesser
threshold.
   exit
exit
```

N7 Interface Outbound

Use the following commands to configure alerts related to an outbound N7 interface.

```
alerts rules group PCFSvcStatus
   interval-seconds 300
   rule PCFN27outbound
   expression sum(increase(outgoing_request_total{interface_name
=\”N7\”,response_status=~\”2..\”}[5m])) /sum(increase(outgoing_request_total{interface_name
 =\”N7\”}[5m])) <0.90
   severity major
   type Communications Alarm
   annotation summary
  value This alert is fired when the percentage of Success N7 responses received is lesser
 threshold.
   exit
exit
```

N28 Interface Inbound

Use the following commands to configure alerts related to an inbound N28 interface.

```
alerts rules group PCFSvcStatus
   interval-seconds 300
   rule PCFN28Inbound
```

```
   expression
sum(increase(inbound_request_total{interface_name=\"N28\",response_status=~\"2..\"}[5m]))
/sum(increase(inbound_request_total{interface_name =\"N28\"}[5m])) <0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the percentage of Success N28 responses sent is lesser
 threshold.
   exit
exit
```

N28 Interface Outbound

Use the following commands to configure alerts related to an outbound N28 interface.

```
alerts rules group PCFSvcStatus
   interval-seconds 300
   rule PCFN28outbound
   expression sum(increase(outgoing_request_total{interface_name
=\"N28\",response_status=~\"2..\"}[5m])) /sum(increase(outgoing_request_total{interface_name
 =\"N28\"}[5m])) <0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the percentage of Success N28 responses received is
lesser threshold.
   exit
exit
```

Diameter Rx Interface Inbound

Use the following commands to configure alerts related to an inbound Diameter Rx interface.

```
alerts rules group PCFSvcStatus
   interval-seconds 300
   rule PCFNRxInbound
   expression
sum(increase(diameter_responses_total{command_code=\"AAA|STA\",response_status=~\"2001\"}[5m]))
 /sum(diameter_responses_total(outgoing_request_total{command_code=\"A AA|STA\"}[5m])) <
0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the percentage of Success Rx responses Send is lesser
threshold.
   exit
exit
```

Diameter Rx Interface Outbound

Use the following commands to configure alerts related to an outbound Diameter Rx interface.

```
alerts rules group PCFSvcStatus
   interval-seconds 300
   rule PCFNRxOutbound
   expression
```

```
sum(increase(diameter_responses_total{command_code=\"RAA|ASA\",response_status=~\"2001\"}[5m]))
 /sum(diameter_responses_total(outgoing_request_total{command_code=\"AAA|STA\"}[5m])) <
0.90
   severity major
   type Communications Alarm
   annotation summary
  value This alert is fired when the percentage of Success Rx responses received is lesser
 threshold.
   exit
exit
```

Process level alerts

CDL Endpoint Down

Use these commands to configure alerts related to CDL endpoint down.

```
alerts rules group cdl-ep-change
   rule pod-down
   expression up{pod=~'cdl-ep.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value CDL EP Pod Down
   exit
exit
```

CDL Slot State Change

Use these commands to configure alerts related to CDL slot state change.

```
alerts rules group cdl-slot-change
   rule pod-down
   expression up{pod=\"cdl-slot-session-c1-m1-0\"} == 0
   severity major
   type Equipment Alarm
   annotation description
   value CDL Pod Slot Change
   exit
exit
```

Diameter Endpoint State Change

Use these commands to configure alerts related to Diameter endpoint state change.

```
alerts rules group diamter-ep-change
   rule pod-down
   expression up{pod=~'diameter-ep.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
```

```
   value Diameter EP Change
   exit
exit
```

ETCD State Change

Use these commands to configure alerts related to etcd state change.

```
alerts rules group ep-mapping-change
   rule pod-down
   expression up{pod=~'etcd-pcf.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value EP Mapping Change
   exit
exit
```

Grafana Dashboard State Change

Use these commands to configure alerts related to Grafana dashboard state change.

```
alerts rules group grafana-dashboard-change
   rule pod-down
   expression up{pod=~'grafana-dashboard.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value Grafana Dashboard Change
   exit
exit
```

Kafka State Change

Use these commands to configure alerts related to Kafka state change.

```
alerts rules group kafka-change
   rule pod-down
   expression up{pod=~'kafka.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value Kafka Changed
   exit
exit
```

LDAP Endpoint State Change

Use these commands to configure alerts related to LDAP endpoint state change.

```
alerts rules group ldap-change
   rule pod-down
   expression up{pod=~'ldap-pcf.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value LDAP Pod Changed
   exit
exit
```

CPC Engine State Change

Use these commands to configure alerts related to CPC Engine state change.

```
alerts rules group pcf-engine-change
   rule pod-down
   expression up{pod=~'pcf-engine-pcf.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value CPC Engine Changed
   exit
exit
```

REST Endpoint State Change

Use these commands to configure alerts related to REST endpoint state change.

```
alerts rules group pcf-rest-ep-change
   rule pod-down
   expression up{pod=~'pcf-rest-ep.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value CPC Rest EP Change
   exit
exit
```

Call Flow procedure alerts

LDAP Query Request

Use these commands to configure alerts related to LDAP Query Request.

```
alerts rules group PCFProcStatus
   interval-seconds 300
   rule LDAPQuery
  expression sum(increase(message_total{type=~\".*_ldap_query\", status=\"success\"}[5m]))
 /sum(increase(message_total{type=~\".*_ldap_query\"}[5m])) < 0.90
   severity major
```

```
   type Communications Alarm
   annotation summary
   value This alert is fired when the success percentage of ldap query request is lesser
 threshold.
   exit
exit
```

LDAP Modify Request

Use these commands to configure alerts related to LDAP Modify Request.

```
alerts rules group PCFProcStatus
   interval-seconds 300
   rule LDAPModify
   expression sum(increase(message_total{component=\"ldap-ep\", type=~\".*_ldap_modify\",
 status=\"success\"}[5m])) / sum(increase(message_total{component=\"ldap-ep\",
type=~\".*_ldap_modify\"}[5m])) < 0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the success percentage of ldap modify request is lesser
 threshold.
   exit
exit
```

PLF Query Request

Use these commands to configure alerts related to PLF Query Request.

```
alerts rules group PCFProcStatus
   interval-seconds 300
   rule PLFRequest
   expression
sum(increase(message_total{type=~\"ldap_search-res_success\",status=\"success\"}[5m]))
/sum(increase(message_total{type=~\"ldap_search-res_.*\"}[5m])) <0.90
   severity major
   type Communications Alarm
   annotation summary
   value This alert is fired when the success percentage of PLF request is lesser threshold.

   exit
exit
```

NAP Notification Request

Use these commands to configure alerts related to NAP Notification Request.

```
alerts rules group PCFProcStatus
   interval-seconds 300
   rule NAPNotification
   expression sum(increase(message_total{type=~\"ldap_change-res_success\",
status=\"success\"}[5m])) /sum(increase(message_total{type=~\"ldap_change-res_.*\"}[5m]))
<0.90
   severity major
   type Communications Alarm
```

```
   annotation summary
   value This alert is fired when the success percentage of NAP request is lesser threshold.

   exit
exit
```

System alerts

Disk full alert

Use the following commands to configure alerts related to disk full alert.

```
alerts rules group
   rule node-disk-running-full
   expression node_filesystem_usage > 0.0001
   duration 5m
   severity critical
   type Processing Error Alarm
   annotation disk_full
   value test
   exit
exit
```

VM down alert

Use the following commands to configure alerts related to virtual machine down alert.

```
alerts rules group vm-state-change
   rule vm-down
   expression up{pod=~\"node-expo.*\"} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation summary
   value VM Down
   exit
exit
```

High memory usage

Use the following commands to configure alerts related to high memory usage.

```
alerts rules group memory-util-high
   rule mem-util-high
   expression avg(node_memory_MemAvailable_bytes /node_memory_MemTotal_bytes * 100) by
(hostname) < 20
   duration 1m
   severity critical
   type Processing Error Alarm
   annotation mem_util_high
   value Hig Memory Usage
   exit
exit
```

High disk usage

Use the following commands to configure alerts related to high disk usage alert.

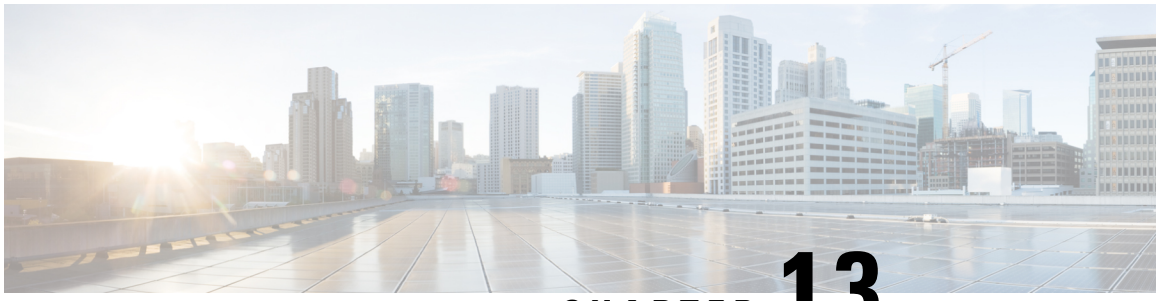
```
alerts rules group disk-util-high
   duration 1m
   rule disk-util-high
   expression avg (node_filesystem_avail_bytes{mountpoint =\"/\"}
/node_filesystem_size_bytes{mountpoint =\"/\"} *100) by (hostname) <20
   severity critical
   type Processing Error Alarm
   annotation description
   value Hig Memory Usage
   exit
exit
```

High CPU usage

Use the following commands to configure alerts related to high CPU usage alert.

```
alerts rules group cpu-util-high
   rule cpu-util-idle
   duration 1m
   expression avg(rate(node_cpu_seconds_total{mode='idle'}[1m])) by (hostname) *100 < 50

   severity critical
   type Processing Error Alarm
   annotation description
   value Hig CPU
   exit
exit
```



CHAPTER 13

Event and System logs

Feature Description

CPC provides a centralized view of the application logs that are consolidated from different containers. The unified view improves the efficiency as you can determine the issue faster instead of accessing the individual containers to view the logs. Collection of logs from the containers is enabled by default.

You can view the logs in the real time and offline mode. The real-time mode captures the current event activity that is performed on the container. In the offline mode, you have the flexibility to access the logs from a remote machine.

Logs are listed based on the timestamp at which they are generated.

How it Works

This section describes how this feature works.

The OAM node hosts the logs which different application containers generate. These containers include the CPC (engine), cpc-rest-ep, policy-builder, diameter-ep, ldap-ep, crd, and unifiedapi.

View the logs

This section describes how to view the consolidated application logs.

To view the consolidated logs, use this command:

kubectl logs -n *namespace* **consolidated-logging-0**

NOTES:

- *namespace* – Specifies the namespace under which CPC is deployed.

Troubleshoot Information

This section provides information for troubleshooting any issues that may arise during the feature operation.

If the logs are not generated in the consolidated-logging-0 pod, then one of the following conditions may be causing the failure. To resolve the issue, make sure that you do the following:

- Verify the status of <*namespace*>-cpc-oam-app helm deployment. To view the configured helm charts and their status, use this command:

  helm list

- Ensure that the gRPC stream appender is enabled by verifying the contents of cps-logback configMap. To verify the contents, use this command:

  kubectl describe configmap -n *namespace* **cps-logback**

- Ensure that the consolidated-logging-0 pod is up and running. To check the pod status, use the following command:

  kubectl describe pod consolidated-logging-0 -n *namespace*

- Verify that the consolidated-logging-0 pod is accessible through the consolidated-logging service. To verify the connection, use the `nc` command.

Configure Centralized CDR Export

Feature description

The Centralized CDR Export feature exporting Call Detail Records (CDRs) in CPC. This feature centralizes the log-based CDR replication and export by forwarding engine-generated CDR logs to the dedicated `pcf-cdr-log` pod. CDR processing and generation remain within the CPC engine, the `pcf-cdr-log` pod manages the subsequent CSV normalization, local file persistence, rotation, compression, and optional secure remote export through FTP.

This design replaces legacy engine-side replication, improving reliability through centralized management, configurable retention policies, and automated retry logic for failed FTP transfers.

CDR Policy Builder configuration

Before the `pcf-cdr-log pod` exports CDRs, configure the CPC engine to identify, process, and structure the data. This configuration establishes the source of truth for CDR generation. The engine handles processing, record calculation, and internal message queue management.

Note If the configuration phase is incomplete or contains errors, the export pod will not receive any data to process.

This section describes the process of preparing the Policy Builder to generate CDRs.

Install the required features on the target engine (`pcf-green`) to enable the reporting plugin.

1. Access the Ops-Center CLI.
2. Execute these commands:

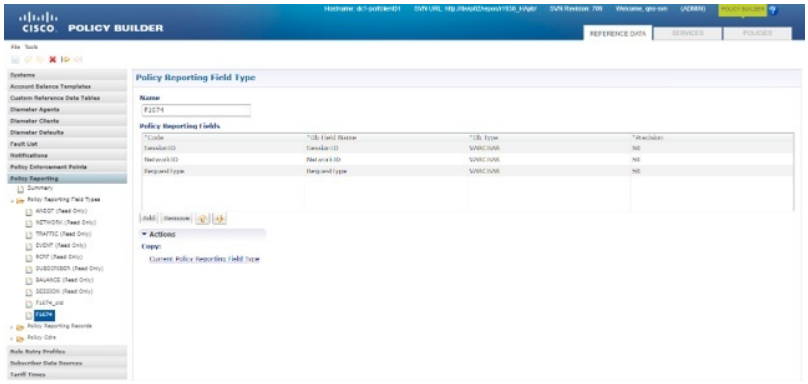
```
engine pcf-green
install-features policy-builder [ com.broadhop.client.feature.policyintel ]
install-features policy-server [ com.broadhop.policyintel.service.feature ]
exit
```

Policy Reporting Field types

Specify the structure for each data attribute you plan to gather.

1. Navigate to **Reference Data** > **Policy Reporting** > **Policy Reporting Field** Types.
2. Create a new field type for required attributes.
3. Specify the **Code**, **DB field name**, and **Data type** (e.g., VARCHAR) for each.

Figure 35: Policy Reporting Field Types configuration

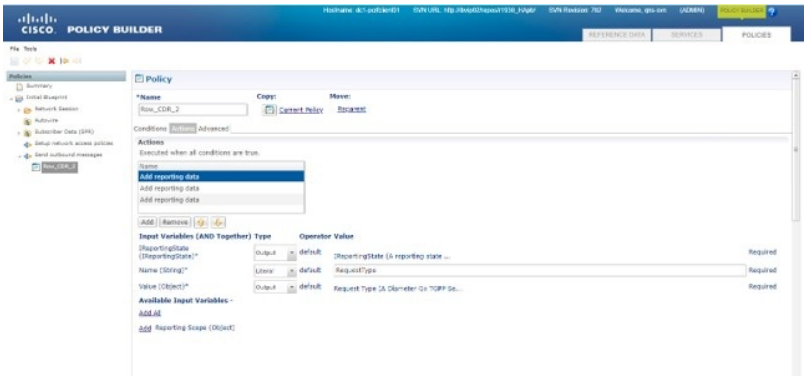


Custom Policy Logic

Define the logic that triggers CDR generation.

1. Add the `sendoutboundmessages` extension point to the **Initial Blueprint**.
2. Create a custom policy (example: `Row_CDR_2`) under **Policies** > **Initial Blueprint** > **Send outbound messages**.
3. Configure the policy **Conditions** (example: verify that a Diameter Gx TGPP session exists).
4. Use the **Add reporting data** action to map session variables to the reporting state.

Figure 36: Add reporting data action

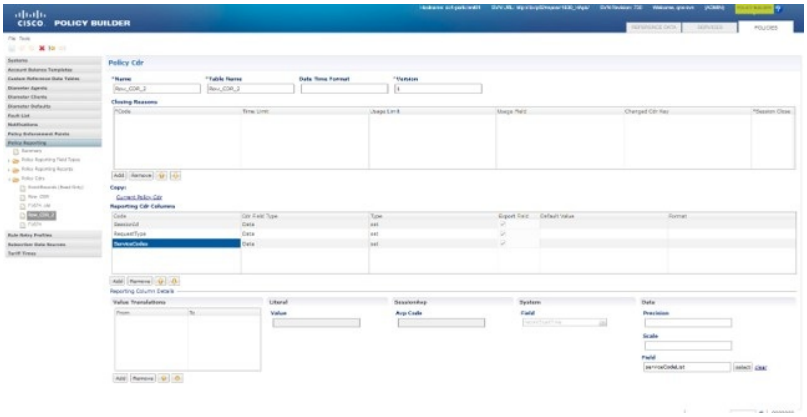


CDR Record Structure

Define the template for the exported records.

1. Navigate to **Reference Data** > **Policy Reporting** > **Policy CDRs**.
2. Create the CDR definition.
3. Enable the **Export** field for each column required in the output. Confirm that these fields correspond to the reporting state variables defined in the custom policy.

Figure 37: Policy CDR record structure



Reporting Server and Plugin Configuration

Link the PB configuration to the reporting engine.

1. Navigate to **Reference Data** > **Systems** > **system-1** > **Plugin Configurations** > **Policy Reporting Configuration**.
2. Set the **DB hosts** to match your environment (example: primary db: `admin-db-0.admi-db`, secondary `db: admin-db-1.admi-db`, port: `27017`).
3. Under **Reporting Server Configuration**, set the **Related CDR** to the definition created in CDR Record Structure.

4. Configure the replication child (CSV or Realtime CSV).
   - Constraint: Only one CSV configuration should be active per reporting server.

Figure 38: Reporting Server Configuration

Configure centralized CDR export

Before you begin

Ensure that you meet these requirements:

- **Access**: Ensure the CPC Ops-Center is accessible.
- Enable Policy Intelligence support in Policy Builder and the Policy Server by running this commands:

```
engine pcf-green
install-features policy-builder [ com.broadhop.client.feature.policyintel ]
install-features policy-server [ com.broadhop.policyintel.service.feature ]
exit
```

- Verify that the CPC engine is configured with the Java Virtual Machine (JVM) property.

```
properties log.cdr.csv
value true
exit
properties disableCdrReplication
value true
exit
```

Use this procedure to enable the CDR export service, manage log rotation, and configure secure remote backups.

Procedure

Step 1 Log in to the CPC Ops-Center CLI.

Step 2 Enable the CDR export service.

```
pcf-cdr enabled true
```

Step 3 Configure the log rotation parameters to manage local storage usage:

```
pcf-cdr log-rotation roll-interval-seconds <seconds>
pcf-cdr log-rotation max-file-size-mb <size_in_mb>
pcf-cdr log-rotation max-final-files <number_of_files>
pcf-cdr log-rotation gzip-final-files true
```

Note
`roll-interval-seconds`: Set to a value greater than 0 to enable time-based rolling; set to 0 to enable size-only rolling (`max-file-size-mb`).

Step 4 Enable and configure the FTP backup settings:

```
pcf-cdr log-backup enabled true
pcf-cdr log-backup ftp-server-user-name <username>
pcf-cdr log-backup ftp-server-user-ip <ip_address>
pcf-cdr log-backup ftp-server-dest-backup-path <remote_path>
pcf-cdr log-backup ftp-server-port 22
pcf-cdr log-backup ftp-server-user-password <password>
pcf-cdr log-backup connect-timeout-seconds 10
pcf-cdr log-backup strict-host-key-checking false
pcf-cdr log-backup poll-interval-seconds 30
pcf-cdr log-backup interval-seconds 0
```

pollIntervalSeconds: staging folder scan frequency (staging, retention, cleanup).

intervalSeconds = FTP transfer throttle (0 = no throttle, meaning attempt upload on every poll).

Note
If roll-interval-seconds >0 time based rolling happens otherwise enables size-only

Step 5 Verify the CDR export service

a) Confirm the `pcf-cdr-log` stateful set is running.

```
kubectl get pod -n pcf pcf-cdr-log-0
```

b) Verify that the system populates the active CDR file.

```
kubectl exec -n pcf pcf-cdr-log-0 -- tail -f /data/pcf-cdr/cdr.csv
```

c) Confirm that the system moves rolled files to the final directory.

```
kubectl exec -n pcf pcf-cdr-log-0 -- ls -ltr /data/pcf-cdr/final
```

Log file management

The `pcf-cdr-log` pod manages log files through active recording, archiving, and remote transfer.

- **Active logs**: The `pcf-cdr-log` pod writes formatted CSV records to `/data/pcf-cdr/cdr.csv`.
- **Archived logs**: Once the file size or interval limit is reached, the system moves the rolled file to `/data/pcf-cdr/final`. If the `gzip-final-files` property is set to `true`, the file is compressed into a gzip format; otherwise, it remains a plain file. The system retains these files based on the `max-final-files` setting.
- **FTP transfers**: If FTP is enabled, the system copies files to `/data/pcf-cdr/stagingFTP` for transfer. Upon successful upload, the system creates an `.uploaded` marker in `/data/pcf-cdr/uploadedFTP`.

Note In the event of persistent FTP transfer failures, the system retains a maximum of 20 files in the `stagingFTP` directory. Any files exceeding this limit are automatically deleted.

Troubleshoot CDR export

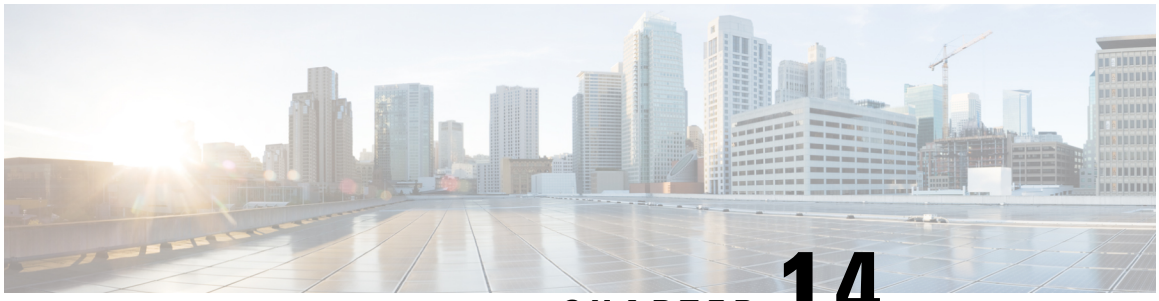
Use this table to identify and resolve common issues encountered during the configuration and operation of the centralized CDR export.

Table 27: Troubleshooting common issues

Issue	Potential cause	Corrective action

No CDR records in `cdr.csv`	The PCF engine is not configured with
	`properties log.cdr.csv`
	`value true`
	`exit`
	.
		Verify the engine JVM properties.

FTP transfer fails	Incorrect credentials or network path.	Check the Ops-Center FTP configuration and network connectivity.
High disk usage	`max-final-files` is set too high.	Adjust `log-rotation max-final-files` in the Ops-Center.



CHAPTER 14

Troubleshooting

CPC deployment issue debugging

This section describes how to debug the issues that may occur when you deploy CPC through the SMI Deployer.

To debug the deployment issues, use the following checklist. If the checklist does not assist you in resolving the issue, analyze the diagnostic data that is available in the form of logs.

Table 28: Troubleshooting Checklist

Task	Resolution
Verify if the Ops Center is refreshing with the latest configurations	Manually verify if the configurations are refreshed. If the Ops Center is not refreshing or displaying the recent changes, then reinstall the helm charts.
Validate if the external IPs and ports are accessible.	Use Telnet or any other application protocol and access the external IP address. This is to confirm that the IP address is accessible. If you are unsure of the IP address, run the following in the Kubernetes service to view the configured external IP addresses and port number: **`kubectl get services -n`** *`namespace`*
Ensure that the IP addresses and ports that are configured for CPC are open in the firewall.	Use the following command to open the ports: **`firewall-cmd –zone=public –add-port=`** *`port`***`/tcp –permanent`**

Task	Resolution
Confirm if CPC connects with the other NFs.	Use the following command on the master node to verify that a healthy connection is available between the NFs: **nc -v** Alternatively, from the proto VM, run the **nc -v** command on the Telnet CLI.
Validate that the successfully deployed helm chart is listed in the helm list.	Use the following steps to determine which helm chart is not listed in the helm list: 1. Run the following on the master node to view the list of deployed helm charts: **helm list** 2. If the helm chart is not found, run the following in the operational mode to view the charts irrespective of their deployment status. **show helm charts** 3. Review the CPC Ops-Center logs to identify the helm chart which has the issue. Depending on the issue, take the appropriate action. Alternatively, you can review the consolidated set of logs, using the following command: **kubectl logs -n namespace consolidated-logging-0**

Refresh the CPC Ops Center

This task refreshes the CPC Ops Center to display the latest configurations and resolve issues with stale data or unexpected responses.

The CPC Ops Center may not consider recent configurations, causing you to observe stale data or not get the expected response. You can refresh the CPC Ops Center using basic and advanced steps.

Before you begin

Perform the advanced steps only when the basic steps do not resolve the issue.

Follow these steps to refresh the CPC Ops Center:

Procedure

Step 1 Run the following to undeploy CPC from the Ops Center:

Example:

```
system mode shutdown
```

Step 2 Use the following to manually purge any pending deployments from the helm:

Example:

```
helm delete --purge helm_chart_name
```

Step 3 From the master node, run the following to delete the configMaps from the namespace where CPC is installed:

Example:

```
kubectl delete cm config_map_name -n namespace
```

Step 4 Delete the product-specific configMaps from the CNEE namespace.

a. Use the following to list the available configMaps:

```
kubectl get configmaps -n namespace
```

From the list, determine the configMap that you want to delete.

b. Run the following to delete the configMap:

```
Kubectl delete configmap configmap_name -n namespace
```

Step 5 Use the following commands to reinstall the helm chart:

Example:

```
helm upgrade –install release name addR/chart_name -f filenames --namespace namespace
```

Once the chart is installed, a new instance of the CPC Ops Center is available.

Step 6 If the basic steps do not resolve the issue, remove the CNEE-ops-center and delete the configMaps from the namespace.

For more information, see the previous steps.

Step 7 Install the CPC Ops Center.

Step 8 If the recent configuration is not rendered because the responsible pods are not in a healthy state, use the following command to view the pod's logs:

Example:

```
kubectl describe pod pod_name -n namespace
```

The recent configuration is not rendered because the responsible pods are not in a healthy state to process the refresh request. To investigate the issue at the pod level, review the pod's state.

Alternatively, you can review the consolidated set of logs, using the following command:

```
kubectl logs -n namespace consolidated-logging-0
```

In the logs, the values in the Status and Ready columns indicate the following:

- If the Status column displays the state as Running, and the Ready column has the same number of containers on both sides of the forward-slash (/), then the pod is healthy and operational. This implies that the issue is at the application level. To investigate the application issue, check the logs of all the containers residing within the pods to detect the issue. Or, log into the container and review the logs.
- If the Status column displays the state as Pending, Waiting, or CrashLoopBackOff, then run the following to review the details such as the messages, reasons, and other relevant information:

  ```
  kubectl describe pod pod_name -n namespace
  ```

- If the Status is init or ContainerCreating, it signifies that the pod is in the process of starting up.
- If the Status is Running, and in the Ready column the number of containers on both sides of forward-slash (/) are different, then the containers have issues.

  Run the following to view the details:

kubectl describe pod *pod_name* **-n** *namespace*

When reviewing the details, if the Ready column has the value as false then it indicates that the corresponding container has issues. Review the associated logs to understand the issue.

- If the Status and Ready columns, and logs of the container do not indicate any issue, then verify that the required ingress or the service that is required to reach the application is up and running.

The CPC Ops Center is refreshed and displays the latest configurations. The issue with stale data or unexpected responses is resolved.

Resolve subscriber not found or primary key not found issues

This task helps you resolve issues that occur when Network Functions (NFs) cannot find subscriber details and send Subscriber Not Found or Primary Key Not Found messages to CPC.

When the NFs cannot find the subscriber details, they send the Subscriber Not Found or Primary Key Not Found to CPC. This section describes how to resolve these issues.

Procedure

Step 1 Analyse the logs of the CPC Engine and REST endpoint pod for the subscriber or primary key related issues.

On the master node, run these command to determine the engine and REST-ep pod.

kubectl logs -n *namespace pod_name*

Step 2 Navigate to the pods and review the subscriber availability status and the subscriber count in the database.

cdl show session *count/summary*

Based on the subscriber's status, take the appropriate action to resolve the issue.

The subscriber or primary key related issues are identified and resolved based on the analysis of logs and subscriber status in the database.

Message routing issues

This section describes how to troubleshoot the message routing issues.

Problem

You may observe a message routing failure when a message from the CPC endpoint incorrectly routes a message from Canary to the CPC Engine. The issue occurs when the message is sent to an incorrect CPC group.

Resolution

The following conditions might be causing the message routing failure. Check for these conditions and correct them, if necessary.

- From the CPC Ops Center, manually verify that the routing rules are configured correctly and they match the incoming traffic.
- Ensure that the Istio proxy is injected in the cpc-rest-ep pod.
- Verify that the virtual services are generated using the **istioctl** command.
- Enable the DEBUG level for com.cisco.cpc.endpoint.routing and review the CPC-rest-ep logs for any issues. Use the following command to enable the DEBUG level:

  ```
  debug logging logger com.cisco.cpc.endpoint.routing level debug
  ```

Troubleshooting information collection

If you encounter issues in your CPC environment, gather and analyse the information associated to the failed action or process. Having this information enables you to detect the component that experiences the failure and resolve the issue faster.

Issues

These table covers the components which might experience an issue, and the logs that contain the information corresponding to the issue.

Table 29: Issues

Issue	Logs

Deployment errors	Review these logs to determine the issue. These logs assist you in identifying the component that may be the source of the error.
	Use the following commands on the master node:
	• View the available pods and review the pod status:
	`kubectl logs -n namespace pod_name`
	Depending on the pod's state, perform the appropriate remediation actions. View the configured helm charts and their status:
	• `helm list`
	• View the helm chart details for the REST endpoint:
	`helm get namespace -CPC-rest-ep`

Communication issues between the NFs	1. On the master node, run these command to identify the pod that is responsible for the communication: `kubectl logs -n namespace pod_name` 2. Use the tcpdump utility to trace the packets.

Issue	Logs
Registration and deregistration issues	Use the following command to review the CPC REST endpoint logs: **helm get** *namespace* **-CPC-rest-ep**
Ops Center issues	Review the pod's log that hosts the Ops Center to determine the issue. **kubectl logs -n** *namespace pod_name* To resolve the issue, if you require the configuration information, then run one of the following commands: **show full-configuration** Or, **show running-config**
Traffic routing issues	To view the traffic routing-specific logs, use the following configuration: **kubectl get pod -o yaml -n** *namespace CPC-rest-ep pod_name* **istioctl get virtualservice -n** *namespace* **-o yaml** **istioctl get destinationrules -n** *namespace* **-o yaml** Also, review the logs of the following pods: • CPC-REST-ep instance • CPC-engine instance • Datastore or Session DB
Subscriber issues	Review the logs associated to the CPC Engine and REST endpoint to determine the issue. For additional information about the subscriber availability status and the subscriber count in the database, run the following command: **cdl show session** *count/summary*

Alerts

Alerts are notification messages that are generated when incidents requiring your attention or response occur. Review the historical and active alerts to determine the issue.

Alerts for CPC are generated through the CEE utility. To view these alerts, run the following command in the CEE Ops Center:

For active alerts:

show alerts active

For historical alerts:

show alerts history

Note You must have appropriate permission to view the alert details.

Interface ERROR codes

This section describes the codes that CPC reports for the interface errors.

Interface codes are generated as part of the logs or captured in the statistics.

These tables describe the ERROR and the corresponding codes:

Table 30: N7 error codes

ERROR	ERROR Code	Description
USER_UNKNOWN	400 Bad REQUEST	The HTTP REQUEST is REJECTED because the end USER who is specified in the REQUEST is UNKNOWN to the CPC.
ERROR_INITIAL_PARAMETERS	400 Bad REQUEST	The HTTP REQUEST is REJECTED. This ERROR is reported when the set of session or subscriber information which CPC requires for a rule selection is incomplete, erroneous, or UNAVAILABLE for decision making. For example, QoS, RAT type, and subscriber information.
ERROR_TRIGGER_EVENT	400 Bad REQUEST	The HTTP REQUEST is REJECTED because the set of session information sends a message that originated due to a TRIGGER is incoherent with the previous set of session information for the same session. For example, TRIGGER met was RAT changed, and the RAT notified is the same as before.
TRAFFIC_MAPPING_INFO_REJECTED	403 Forbidden	The HTTP REQUEST is REJECTED because the CPC doesn't accept one or more of the TRAFFIC mappings filters provided by the SMF in a PCC REQUEST.
ERROR_CONFLICTING_REQUEST	403 Forbidden	The HTTP REQUEST is REJECTED because the CPC can't accept the UE-initiated resource REQUEST as a network initiated resource allocation is already in-progress. This resource allocation has packet filters that cover the packet filters in the received UE-initiated resource REQUEST. The SMF rejects the attempt for a UE-initiated resource REQUEST.

ERROR	ERROR Code	Description
POLICY_CONTEXT_DENIED	403 Forbidden	The HTTP REQUEST is REJECTED because the CPC doesn't accept the SMF REQUEST due to operator policies and local configuration.

Table 31: N28 error codes

ERROR	ERROR Code	Description
USER_UNKNOWN	400 Bad REQUEST	The subscriber that is specified in the REQUEST isn't known at the CHF and the subscription can't be created.
NO_AVAILABLE_POLICY_COUNTERS	400 Bad REQUEST	There are NO POLICY COUNTERS AVAILABLE for the subscriber at the CHF.

Note The generic ERROR codes are applicable for all the network interfaces.

Table 32: Generic error codes

ERROR	ERROR Code	Description
TIMEOUT	408 REQUEST TIMEOUT	The HTTP REQUEST to the server took longer than the period the server is configured to wait.
OVERLOAD	429 Too Many Requests	The server has received too many consecutive requests to process within a short interval.
INTERNAL_ERROR	500 INTERNAL Server ERROR	The server has encountered an unprecedented condition, which does not have an appropriate message.
SERVICE_UNAVAILABLE	503 SERVICE UNAVAILABLE	The server cannot process the REQUEST because it is either, overloaded or is UNAVAILABLE due to scheduled maintenance. This is a transient state.

Forward logs to the Splunk Server

Enable CPC to forward logs to a Splunk server which provides index-based search capability for monitoring purposes.

Splunk is a third-party monitoring application that stores the log files and provides index-based search capability. You can configure CPC to send the logs securely to a Splunk server which could be an external server.

Important The Splunk server is a third-party component. Cisco does not take the responsibility of installing, configuring, or maintaining this server.

Follow these steps to forward the logs to the Splunk server:

Procedure

Configure the Splunk forwarding settings using these configuration.

Example:

```
config
  debug splunk
    batch-count no_events_batch
    batch-interval-msbatch_interval_ms
    batch-size-bytes batch_size
    hec-tokenhec_token
    hec-url hec_url
    end
```

These is an example configuration:

```
configure
 debug splunk hec-url https://splunk.10.86.73.80.nip.io:8088
 debug splunk hec-token 68a81ab4-eae9-4361-92ea-b948f31d26ef
 debug splunk batch-interval-ms 100
 debug splunk batch-count 10
 debug splunk batch-size-bytes 102400
 end
```

NOTES:

- **debug splunk**—Enters the configuration debug mode.
- **batch-count** *no_events_batch*—Specify the maximum number of events to be sent in each batch.
- **batch-interval-ms** *batch_interval_ms*—Specify the interval in milliseconds at which a batch event is sent.
- **batch-size-bytes** *batch_size*—Specify the maximum size in bytes of each batch of events.
- **HEC-token** *hec_token*—Specify the HTTP Event Collector (HEC) token for the Splunk server.

- **HEC-url** *hec_url*—Specify the protocol, hostname, and HTTP Event Collector port of the Splunk server. The default port is 8088.

CPC is configured to forward logs to the Splunk server using the specified parameters for batch processing and HTTP Event Collector settings.

Pods stop running when CPC is upgraded through the Rolling Upgrade process

This section describes how to ensure that the pods are running when CPC is upgraded.

Problem

When the CPC version is upgraded to the subsequent available version, some pods such as CRD and Policy Engine stop running.

Resolution

Whenever you configure CPC ensure that you configure these parameters:

- **db global-settings db-replica** *replica_count*
- **db spr shard-count** *shard_count*
- **rest-endpoint ips** *ip_address1, ip_address2, ip_address3*
- **rest-endpoint port** *port_number*
- **engine** *engine_name*

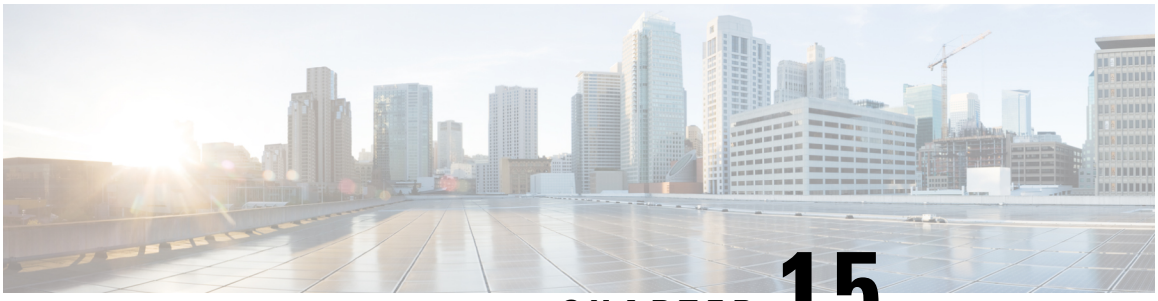
  replicas *replica_count*

  unified-api-replicas *api_replica_count*

  subversion-run-url *repository_url*

  subversion-config-url *configuration_url*

  tracing-service-name *service_name*
- **service-registration profile locality** *profile_name*
- **service-registration profile plmn-list [mcc mnc]**
- **service-registration profile snssais [sst sd]**



CHAPTER 15

Sample configurations

Sample configuration file

The following is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

Important The mandatory parameters are required to ensure that the critical pods such as CRD and Policy Engine are in the running state.

```
config
datastore primary-endpoint connection-settings keep-alive keep-alive-time-ms 200
datastore primary-endpoint connection-settings channel count 4
datastore primary-endpoint connection-settings timeout-ms 500
datastore external-endpoints datastore
 connection-settings keep-alive keep-alive-time-ms 200
 connection-settings channel count 3
 connection-settings timeout-ms 500
exit
ldap replicas 2
ldap server-set USD
 search-user dn cn=sdcUser,dc=C-NTDB
 search-user password $8$yx0jELXTK0f7CJO2XklpJx+CpCUIX13B9C5oQ4NEnaI=
 health-check interval-ms 5000
 health-check dn cn=sdcUser,dc=C-NTDB
 health-check filter msisdn=918369110173
 health-check attributes napCustType
 initial-connections                              10
 max-connections                                  10
 retry-count                                      2
 retry-timer-ms                                   100
 max-failover-connection-age-ms                   60000
 binds-per-second                                 0.2
 number-consecutive-timeouts-for-bad-connection -1
 missing-attribute-result-code                    32
 connection 192.0.2.18 389
  priority        400
  connection-rule ROUND_ROBIN
  auto-reconnect  true
  timeout-ms      200
```

```
  bind-timeout-ms 3000
 exit
 connection 192.0.2.18 390
  priority        400
  connection-rule ROUND_ROBIN
  auto-reconnect  true
  timeout-ms      200
  bind-timeout-ms 3000
 exit
 connection 192.0.2.18 391
  priority        400
  connection-rule ROUND_ROBIN
  auto-reconnect  true
  timeout-ms      200
  bind-timeout-ms 3000
 exit
exit
//This is a mandatory parameter
db global-settings db-replica 3
//This is a mandatory parameter
db global-settings volume-storage-class local
db spr shard-count 1
db balance shard-count 1
debug tracing type DISABLED
debug logging default-level error
debug logging logger com.broadhop
 level warn
exit
debug logging logger com.broadhop.custrefdata.impl.dao.GenericDao
 level error
exit
debug logging logger com.broadhop.diameter2.policy.endpoints
 level error
exit
debug logging logger com.broadhop.ldap
 level error
exit
debug logging logger com.broadhop.microservices.control
 level error
exit
debug logging logger com.broadhop.utilities.queue.redis
 level error
exit
debug logging logger com.cisco
 level warn
exit
debug logging logger com.cisco.diameter
 level error
exit
debug logging logger com.cisco.diameter.endpoint
 level error
exit
debug logging logger com.cisco.pcf
 level debug
exit
debug logging logger com.cisco.pcf.endpoint.client
 level error

exit
debug logging logger com.cisco.pcf.endpoint.client.Http2JettyRequestAsync
 level error
exit
debug logging logger com.cisco.pcf.ldapserver
 level warn
```

```
exit
debug logging logger com.cisco.pcf.nf.cache.NfCache
 level warn
exit
debug logging logger io.prometheus.client
 level error
exit
debug logging logger policy.engine
 level debug
exit
debug logging logger rest.message
 level warn
exit
features patching ingress-enabled true
diameter settings timeouts-ms dpa 5000
diameter application rx
 application-id   16777236
 tgpp-application true
 vendor           [ 10415 ]
exit
diameter group rx-protocol-1
 endpoint-group rx-1
  priority 50
  endpoint rx-endpoint-1
   priority 50
   enabled true
   host-name pcf-diameter-endpoint
   port 3868
   transport-protocol tcp
   application [ rx ]
   timers connection-timeout 30
   timers watchdog 30
   timers reconnect-timeout 30
  exit
 exit
exit
diameter service rx-service
 bind 0.0.0.0:3868
 group rx-protocol-1
 application rx
 transport-protocol tcp
exit
aaa authentication users user admin
 uid        1117
 gid        100
 password   $1$xyz$abcdefghijklmnopqrstuvwxyz
 ssh_keydir /var/confd/homes/admin/.ssh
 homedir    /var/confd/homes/admin
exit
nacm groups group admin
 user-name [ admin ]
exit
nacm groups group policy-admin
 user-name [ admin ]
exit
nacm rule-list admin
 group [ admin ]
 rule any-access
  action permit
 exit
exit
nacm rule-list confd-api-manager
 group [ confd-api-manager ]
 rule any-access
```

```
  action permit
 exit
exit
nacm rule-list ops-center-security
 group [ * ]
 rule change-self-password
  module-name       ops-center-security
  path              /smiuser/change-self-password
  access-operations exec
  action            permit
 exit
 rule smiuser
  module-name       ops-center-security
  path              /smiuser
  access-operations exec
  action            deny
 exit
exit
```