

Cisco Ultra Cloud Core CPC AAA- Configuration and Administration Guide, Release 2026.04

First Published: 2026-11-05

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2026 Cisco Systems, Inc. All rights reserved.



CONTENTS

Full Cisco Trademarks with Software License ?

PREFACE

About this Guide xix

Conventions Used xix

Contacting Customer Support xx

CHAPTER 1

CPC Overview 1

CPC overview 1

Components 3

Platform infrastructure upgrade 4

License information 5

CHAPTER 2

Deploy and Configure cnAAA through Ops Center 7

Introduction 7

Base cnAAA Configuration 8

Deploy and Validate cnAAA Software 8

Run the SMI Cluster Manager 8

Deploy the cnAAA 9

Stage the cnAAA Image 9

Trigger cnAAA 10

Configure Cluster Manager for Dual Stack Environment 11

Monitor the Deployment of cnAAA 13

Pod affinity relaxation for maximum resource utilization 14

Disable the optional PODs through Ops Center configuration 16

Sample configuration file 17

Access the cnAAA Ops-Center 27

cnAAA Health Check 27
Verify the Deployment 27
View the Pod Details 27
Verify the Helm Status 30
Verify the Pods 30
Centralized system information command for cnAAA 31
Check the PB, CRD, Ops-Center Configuration 36
Post deployment verification steps 38
Automatic site failover 39
Automatic site failover 39
system-failover command 40
Automatic site failover workflow 41
Configure automatic site failover 42
Automatic site failover trigger scenarios 44

CHAPTER 3 **User and Group Management** 49
Introduction 49
Add a user from Ops Center CLI 49
Delete a user from Ops Center CLI 50
Password change from Ops Center CLI 50
Update password length policy from Ops Center CLI 51
Add a user-group from Ops Center CLI 51
Delete a user-group from Ops Center CLI 52
Assign a user to a user-group 52
Unassign a user from a user-group 53
User privileges and access control in Ops-Center 53

CHAPTER 4 **Cisco Common Data Layer** 55
Feature History 55
Feature Description 55
Limitations 55
How the CDL Works 56
Architecture 56
Call Flows 57

CDL endpoint failure 57
GR Call Flows 58
Configure CDL through Ops Center 60
Configure the CDL session database and defining the base configuration 61
Configure Kafka in CDL 62
Configure Zookeeper in CDL 63
Sample Configuration 64
Configure the CDL Endpoints 64
Configure the External Services 65
Associate the datastore with the CDL endpoint service 65

CHAPTER 5 **Software Upgrade Qualification** 67
Introduction 67
Deploy and validate cnAAA software 68
Run the SMI cluster manager 68
Prerequisites 69
cnAAA Health Check 69
Prepare for upgrade 71
Backup SVN, Policy, and CRD data 72
Auto-backup of PB,CRD and Ops Center configurations 73
Feature History 73
Overview 73
Auto-backup procedure 73
Manual backup CEE and cnAAA Ops-Center configuration 74
Upgrade the cnAAA 75
Stage a new cnAAA image 76
Trigger the rolling software upgrade 76
Monitor the cnAAA Upgrade 78
Verify the Helm Status 79
Verify the Pods 81
Rollback the upgrade 81
Reload cnAAA Ops Center configuration 81
Update cnAAA Ops Center configuration 82
Restore the configuration from backup 82

CHAPTER 6 **Subscriber Profile Repository** 85

Support for Mongo and CDL 85

Feature History 85

Overview 86

Configure of the SPR MongoDB replica set 86

Configure Policy Builder and remote database with USuM configuration 86

Configure the MongoDB tuning 87

Configure Mongo Replica Sets 87

Verification 88

MongoDB authentication 88

Introduction to MongoDB authentication 88

MongoDB authentication process 89

Enable database authentication 89

MongoDB backup user privileges 91

Configure Backup-Only user accounts 92

Verify Backup-Only user account 94

Subscriber migration to multiple SPR database replica sets 95

Overview 95

Data Migration from CPS 7.5 to CPC 95

Replica set for admin DB similar to SPR for sharing across clusters 97

Feature History 97

Overview 97

Label nodes for the admin DB replica set configuration 97

Configure admin DB Replica Sets 98

Configure CRD API properties in Ops Center 98

Policy Builder configuration 99

Support for multiple SPR replica sets 100

Feature History 100

Overview 100

Configure Subscriber Database Replica Sets 100

MongoDB replica set auto-recovery 102

Configure the Network ID in API Router 103

Configure Ops Center for multiple SPR replica sets 105

Multiple arbiter qualification for SCDB 108
Feature History 108
Overview 108
Ops-Center Configuration for multiple arbiters in a Replica Set 108
Consistent replica set routing in apiRouter under high TPS 110
Feature History 110
Overview 110
How replica set routings in apiRouter under high TPS work 111
Sample Bulk provision SOAP requests 111
Retrieve the KPIs for multiple subscriber CRUD operations 113
Configurable MongoDB storage size 114
Configure MongoDB storage size 114
Auto scheduler backup for MongoDB 116
Configure the MongoDB replica set 116
Configure global backup settings 117
Perform on-demand backups and monitoring 117
Restore the SCDB SPR backup 118
Troubleshoot the local backup safety net 119
Subscriber authentication and Called-Station-ID uniqueness 119
Deployment requirements for Called-Station-ID features 120
Credential Index uniqueness enforcement 121
Credential Index records and configuration parameters 121
Configure global Called-Station-ID uniqueness 122
Populate the Credential Index for existing subscribers 123
Disable global Called-Station-ID uniqueness 126
Called-Station-ID authentication rules 126
Called-Station-ID authentication 127
Configure the Subscriber Authentication policy 127
Configure Called-Station-ID rejection handling 128

CHAPTER 7 **Subscriber migration from CPS 7.5 to cnAAA** 129

Subscriber migration from CPS 7.5 to cnAAA (Solution 1) 129
Feature History 129
How subscriber migration works 129

HQoS Turbo Plan Rollout for Subscribers 130
Configure subscriber migration 132
Policy Builder Configuration for USum 133
Logs available for migration 134
Subscriber migration from CPS 7.5 to cnAAA (Solution 2) (SOAP Kafka Relay) 135
Feature History 135
Operational logic and monitoring 135
Configure subscriber migration parameters in Ops-Center 136
Alerts 138
DLQ processor script 138

CHAPTER 8 **RADIUS Endpoint 141**
Feature Summary and Revision History 141
Feature Description 141
Overview 142
Configure the node for the RADIUS Endpoint Pod 142
Configure the RADIUS initialization delay 143
RADIUS Endpoint 144
RADIUS call flow support in CPC 144
Handling CoA mismatch on BNG SRG switchover 147
IPv6 Support 150
Feature History 150
Overview 150
RADIUS Endpoint support in IPv6 150
Mongo Replica Sets configuration 151
Ops-Center configuration for enabling IPv6 in RADIUS Endpoint 151
Dual Stack Support 152
Configure Dual Stack in cnAAA Ops Center 152
Ops Center configuration for enabling an external SNMP server 153
Overview 153
Configure cnAAA Ops-Center for IPv6 154
Configure Geo-redundancy with CDL over IPv6 157
Configure Geo-redundancy with Mongo for SPR or Admin DB over IPv6 158
Configure SNMP alerts through IPv6 address 162

Dual-Stack deployment principles 163
IPv4 and IPv6 forwarding validation 163
Cloud native CPS application on CNDP platform 164
Feature History 164
Overview 164
Configure the RADIUS Endpoint in cnAAA using Ops-Center 164
Verification of Radius Endpoint configuration in cnAAA 165
Solution for Service Mismatch 166
Feature History 166
Overview 166
CoA back-off retry in Ops-Center 166
Ops-Center configuration 167
Policy Builder configuration 167
CoA Request Retry Mechanism 168
Configure gRPC timeout parameters 172
Handling service mismatch between cnAAA and BNG 172
CoA back-off retry in Ops-Center 172
Ops-Center configuration 173
Policy Builder configuration 174
CoA Request Retry Mechanism 175
Configure gRPC timeout parameters 178
Consolidation of CoA processing in a RADIUS endpoint for a given BNG 179
Feature History 179
Overview 179
BNG IP configuration in Policy Builder 179
BNG IP Configuration in Ops Center 180
CoA Request Scenarios in cnAAA 181
Combined multi-CoA support 181
CoA Request and Response Metrics 182
User Plane IP (vBNG) support 183
CoA throttling based on User Plane IP 184
Troubleshoot 184
Virtual NAS-IP label for CoA KPIs 185
Feature description 185

Monitor KPI updates 185
Verify traffic processing 186
Troubleshoot 186
CoA packet authenticator generation 187
Feature description 187
Configure the CoA packet authenticators 187
Monitoring 189
Troubleshoot 189
Circle code labels for RADIUS KPIs 190
Circle code labels for RADIUS KPIs 190
Monitoring tool configuration and verification 190

CHAPTER 9 **Persistent Storage for Policy Configuration** 193
Overview 193
How it works 193
Configure Persistent Storage 194
Enable Support for Persistent Storage 194
Assign Persistent Storage 194
Configure the Restore Capability 195

CHAPTER 10 **Managing Custom Reference Data** 197
Feature Description 197
Configuration support for importing CRD 197
Backup the existing SVN repository 198
Backup the existing CRD 198
Remove the existing CRD from MongoDB 199
Mongo Replica Sets configuration 199
Ops-Center configuration for enabling IPv6 in MongoDB Replica Sets 199
Import and publish the new CRD schema 200
Service barring configuration 203
Import the new CRD table 208
Bulk update for CRD table 209
CRD REST API for OLT NEID and loopback addition 211
Configure CRD REST API (Add/Update/Delete/Query) 211

CRD refresh interval configuration and performance improvements in CRD functionality 214
Feature History 214
Overview 214
Configure CRD refresh interval with Ops-Center 214
Configure DB read preference in Policy Builder 215

CHAPTER 11 Message Prioritization and Overload Handling 217
Overview 217
How it Works 217
Configure inbound message overload handling 218
RADIUS Configuration 218
Overload protection for RADIUS endpoint 220
Feature History 220
Overview 221
Global overload protection in the RADIUS endpoint 221
Configure RADIUS endpoint message rate limits 221
Configure the message rate limit for global protection of RADIUS endpoint 222
CoA throttle support 222
OAM Support 223
Bulk Statistics support 223

CHAPTER 12 RADIUS Peer Load Rebalancing 225
Feature Summary and Revision History 225
Summary Data 225
Feature History 225
Feature Description 225
ULB integration on RADIUS Endpoint 226
Feature History 226
Overview 226
Prerequisites 226
Install and configure ULB for cnAAA 226
Ops Center configuration to enable ULB on RADIUS Endpoint 230

CHAPTER 13 Pods and Services 235

Feature Summary and Revision History 235
Summary Data 235
Feature History 235
Feature Description 236
Pods 237
Services 239
Limitations 241
Configuration Support for Pods and Services 241
Associate Pods to the Nodes 241
View the Pod details and status 242
States 243
Grafana dashboard files 243
cnAAA Grafana dashboards 244

CHAPTER 14 **Advanced Tuning Parameters** 245
Feature Summary and Revision History 245
Summary Data 245
Feature History 245
Feature Description 246
Configuration support for the Advanced Tuning parameters 246
On or Off configuration to enable or disable the features 246
Configure Ops-Center CLI 247
List of features and controls 247
Threading configuration for HTTP2 outgoing requests 252
Istio resource control configuration 252
Redis password configuration 253
Network Slice access control 253
OAM Support 253
Bulk statistics support 253
Radius engine CPU configuration 254
Dynamic resource configuration for engine and radius pods 254
Configure CPU and memory resources for engine and radius pods 254
Operational considerations 255
KPI scheduler 256

Feature description 256
Verify the leader election mechanism 256

CHAPTER 15 cnAAA application based alerts 259
Feature Summary and Revision History 259
Summary Data 259
Feature History 259
Feature Description 260
How it Works 260
Configure alert rules 260
View alert logger 261
Sample alerts configuration 263
Process-Level alerts 263
Call Flow procedure alerts 265
System alerts 266
Support notifications for system and application alarms 268
Revision History 268
Overview 268
System Alarms 268
Monitoring alarm support for SVN repository 270
Application alarms 272
Critical Alert to be Raised RADIUS Pod not Sending Access-Accept 275
Configuration 277
Policy configuration counters 278
Check active alerts 279
Add Hostname in SNMP Traps 281
Configure SNMP trap alerts with hostname 281

CHAPTER 16 Event and System logs 285
Consolidated application logs 285
Summary Data 285
Feature History 286
Feature Description 286
How it Works 286

View the logs 286
Troubleshoot Information 286
Consolidated AAA log files 287
Accounting mismatch with retry and failure logging mechanism to OCS 288
System and Syslog activity logging 288
Search Facility 292
How the log search utility works 292
Log forwarder utility 292
Use the log search utility 293
Configure package level logging 294
Separation of accounting retry messages logging for OCS and PMZ 295
Feature description 295
How this feature works 295
Configuration 296
Use cases 296
Enhanced MongoDB pod log for debugging 301
Feature history 301
Supplementary log forwarding for cnAAA 303
Feature history 303
Configure log forwarding mechanism in Ops-Center 304
Configurable log size and PVC for consolidated logs 307
Feature history 307
Configure log management parameters in Ops-Center 309
Log forwarding to external servers 312
Feature History 312
Configure log forwarding in the CEE and CPC Ops-Centers 312
Known limitations 314
Grafana database monitoring dashboard 315
Grafana dashboard files 315
Audit and logging enhancements 316
Overview 316
Configure CLMS log forwarding 318
Verify audit logging 319
Log noise reduction 319

CHAPTER 17 **Policy Builder Overview** 321

Overview 321
Reference Data 322
Services 323
Feature Summary and Revision History 323
Summary Data 323
Feature History 323
Feature Description 323
Service 323
Adding a Service 324
Service configuration 324
Use Case Templates 325
Configure the Use Case Template 325
Generic service configuration 326
Common Parameters 326
Policies 332
Summary of Policy Tab Capabilities 333
Advantages 333
Considerations 333
Add a System 333
Access the Policy Builder 334
URL to access interface 335
Add and update the services, validation 335
Create a new RADIUS Service Template 337
Policy Enforcement Points 338
Policy Enforcement Point Tree 338
ASR9K PEP Configuration 339
BNG addition in Policy Builder through external API 340
Sort and search BNG IPs in Policy Builder 345
Feature history 345
Advantages 346
UI changes 348
Feature History 348

Unresolved errors display during publish 348
Manage OCS and BNG configurations with single API 350
Feature description 350
Configure the Ops-Center properties for API access 350
BNG device additions to the PB and Ops-Center 351
Radius server addition in PB and Ops-Center 356
SOAP API engine rate limiting 362
Configure SOAP API engine rate limiting 362

CHAPTER 18 **Plugin** 365
Overview 365
Threading Configuration 366
RADIUS Configuration 368
RADIUS AAA Proxy Settings 370
ASR9K Configuration 371

CHAPTER 19 **Control Center** 373
Feature History 373
Overview 373
Functionalities 374
Login to the Control Center 374
Find Subscriber 374
Create Subscriber 375
Find Subscriber Session 375
Delete Subscriber 376
Delete Session 376
Service wise user level bifurcation 377
Configure the subscriber bifurcation report 377
Configure report settings 378
Workflow 378
Subscriber search pagination in Control Center 379

CHAPTER 20 **Troubleshooting Information** 381
Feature Summary and Revision History 381

Summary Data 381
Feature History 381
Debugging the cnAAA Deployment Issues 382
Auditd and sysstat service verification 383
How auditd and sysstat verification works 383
SNMP test trap generation facility 385
SNMP test trap generation script 385
Generate predefined SNMP test traps 386
Generate custom SNMP test traps 387
Verify trap reception and interpretation 387
Issue with Refreshing the cnAAA Ops-Center 388
Subscriber Not Found or Primary Key Not Found 389
Called station id report generation 390
Generate a called station report 391
Forwarding logs to the Splunk Server 391
Centralized PCAP collection and merging 393
Configure and run the packet capture utility 394
System health diagnostics 395
Feature overview 395
Run the system health diagnostics 395
Diagnostic command arguments 399
Troubleshoot Kubernetes pods during system upgrade 400
Feature overview 400
Resolve a Unified API access error 404

CHAPTER 21 **Sample cnAAA Configuration** 407
Sample cnAAA Configuration 407
cnAAA GR configurations 416
Sample ULB Configuration 452



About this Guide

- Conventions Used, on page xix
- Contacting Customer Support, on page xx

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**

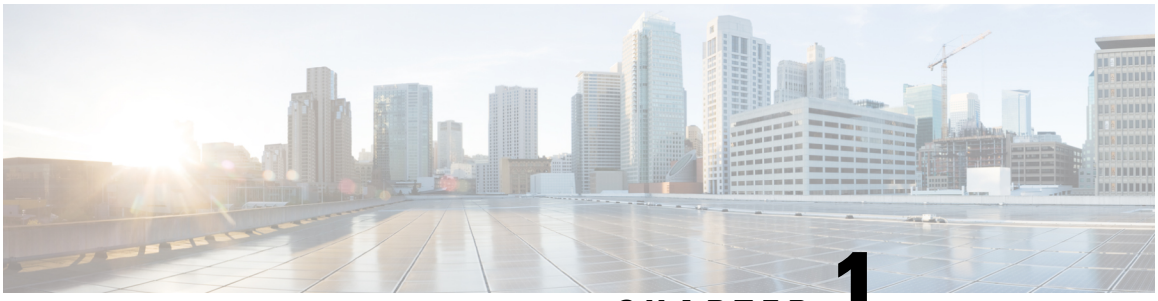
Command Syntax Conventions	Description
{ keyword or *variable* **}**	Required keyword options and variables are those components that are required to be entered as part of the command syntax. Required keyword options and variables are surrounded by grouped braces { }. For example: **sctp-max-data-chunks { 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

CPC Overview

- CPC overview , on page 1
- Platform infrastructure upgrade, on page 4
- License information, on page 5

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 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.

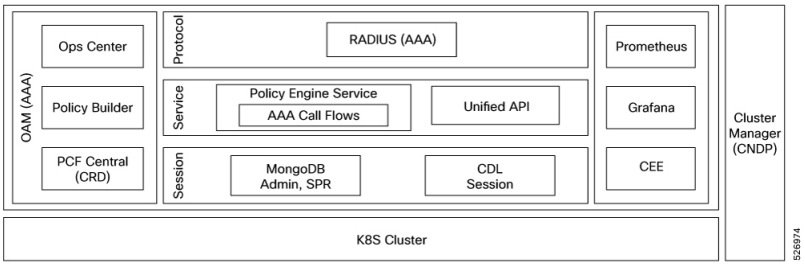
System Composition:

- CPC offers a containerized, cloud-native architecture based on a three-tier microservices framework.
- It supports AAA components using RADIUS 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

Cloud native AAA (cnAAA) is one of the components of Converged Policy and Charging (CPC). cnAAA integrates AAA components within a three-tier micro services framework. This architecture helps you manage tasks related to authentication, authorization, and accounting.

Figure 1: cnAAA architecture



Key Capabilities

The cnAAA implementation provides these capabilities:

- **Protocol Tier**: AAA services use the RADIUS Endpoint protocol. RADIUS endpoints manage protocol interactions. These endpoints forward requests from the Broadband Network Gateway (BNG) to the policy service and relay responses back to the BNG.
- **Service Tier**: The AAA engine handles authentication, authorization, and proxy accounting messages. It manages AAA call flow procedures and selects policies for BNG based on subscriber profiles.
- **Session Tier**: cnAAA uses MongoDB and a CDL endpoint to store data. The system includes an in-memory session store and a subscriber profile database that persists to disk.
- **Operations, Administration, and Maintenance (OAM)**: Ops-Center serves as the console for configuring and administering cnAAA. It supports CLI and RESTCONF API. Ops-Center enhancements include system configuration capabilities. You can configure the number of RADIUS endpoints and fine-tune buffers, queues, and thread pools for AAA services.

  These OAM components are part of the Cisco Cloud Native Data Plane (CNDP) integration. CNDP provides common execution environment services for the cnAAA system.

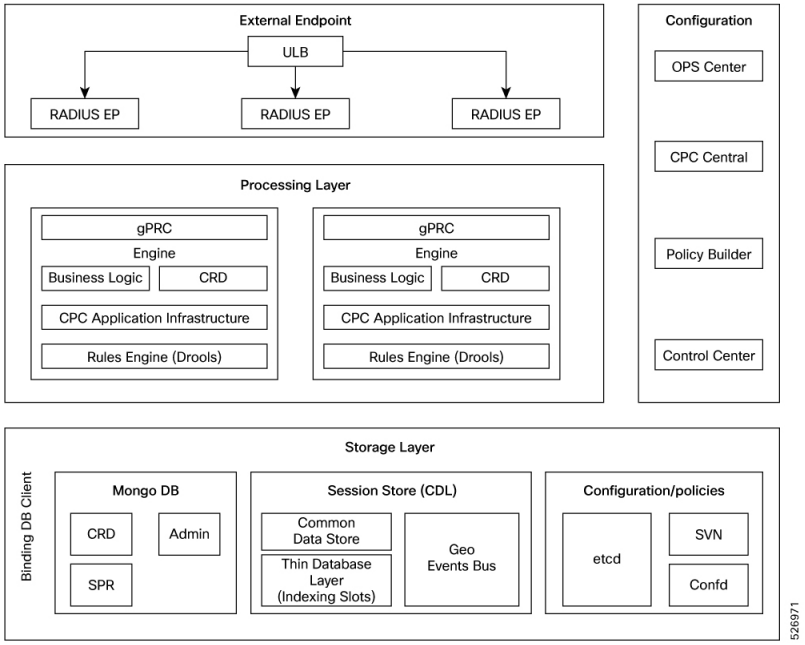
  - **Policy Builder (GUI/API)**: It is enhanced to manage AAA services and configurations. Enhancements include use case templates, service options, and subscriber-triggered groups (STG).
  - **Custom Resource Definitions (CRD)**: Enable data-driven policy implementations through extensible CRD components. The CRD components are extensible, so customers can add new CRD tables as needed.
  - **Integration with Common Execution Environment (CEE)**: The AAA implementation integrates with CEE services. These services include centralized logging, metrics collection using Prometheus, KPI dashboards with Grafana, and alerts.

Note cnAAA provides support exclusively for features related to RADIUS. This document uses the terms CPC and cnAAA interchangeably.

Components

This section provides an overview of the functional components that comprise the cnAAA architecture. It defines the roles of elements in the external endpoint, processing, configuration, and storage layers. This overview illustrates how the system manages network traffic, applies policy logic, and ensures data persistence.

Figure 2: CPC architecture components



The cnAAA comprises of these components:

1. **External endpoint**
    - Unified Load Balancer (ULB) is a Network Function (NF) that manages the distribution of incoming RADIUS traffic to RADIUS endpoints deployed on worker nodes. The ULB ensures high availability and reliability across the network infrastructure.
    - RADIUS-EP: A microservice that provides a channel for inbound and outbound RADIUS messages.
2. **Processing layer**
    - Engine: This component hosts the business logic and drives the rules engine to make policy decisions.
    - gRPC: This framework enables internal processes to communicate and synchronize events.
3. **Configurations**
    - Policy Builder: Allows the configuration of Engine pods, services, and advanced policy rules.
    - CPC Central: A unified GUI that you use to configure the Policy Builder, manage custom reference table data, and access web-based applications such as Grafana and the Control Center.
    - Ops-Center: Allows to configure and manage the applications and pods configuration.
    - etcd: Stores the RADIUS-EP configurations.

4. Storage layer
    - MongoDB: Stores subscriber-specific data and CRD configuration data.
    - Cisco Data Layer (CDL): A dedicated in-memory database used for session persistence.

Platform infrastructure upgrade

Feature Name	Release Information	Description
Platform infrastructure upgrade	2026.02.0	This release upgrades core platform components, including ConfD to version 8.6.2 and Kubernetes to version 1.35, to ensure system stability and security compliance. These updates mitigate known security vulnerabilities and address end-of-life support requirements.

This release upgrades core platform infrastructure components to ensure system stability and security compliance.

The upgrade includes these component updates:

- **ConfD**: Upgraded to version 8.6.2. ConfD manages the system's Yet Another Next Generation (YANG)-based configuration models and provides the CLI and NETCONF interfaces for system administration.
- **Kubernetes**: Upgraded to version 1.35. Kubernetes manages the deployment, lifecycle, and scaling of Cisco Policy Controller (CPC) microservices.

Upgrade drivers

cnAAA requires these upgrades to meet the following lifecycle and security requirements:

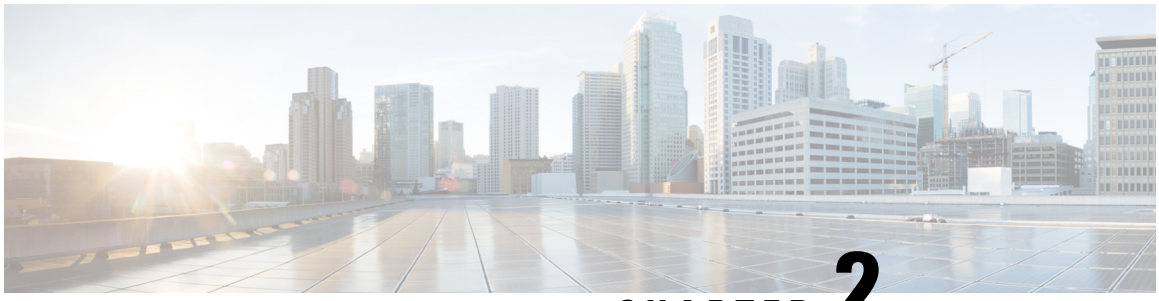
- **End-of-life (EOL) support**: Previous versions of ConfD reached their end-of-life. Upgrading to version 8.6.2 ensures the platform remains supported by the vendor.
- **Security compliance**: The upgrade mitigates identified security vulnerabilities in the software stack, ensuring the platform remains protected against known threats.

Upgrade delivery and impact

- **Delivery mechanism**: This infrastructure upgrade is bundled into the standard CPC software build. No manual installation, configuration, or migration steps are required.
- **Operational impact**: This is a backend-only update. There are no changes to system behavior, user interfaces, or operational workflows. The upgrade is transparent, and system functionality remains consistent with previous releases.

License information

The cnAAA supports Cisco Smart Licensing, a cloud-based licensing model that simplifies the purchase, deployment, and management of Cisco software assets. For more information, refer to the Smart Licensing collection page.



CHAPTER 2

Deploy and Configure cnAAA through Ops Center

- Introduction, on page 7
- Base cnAAA Configuration, on page 8
- Deploy and Validate cnAAA Software, on page 8
- Check the PB, CRD, Ops-Center Configuration, on page 36
- Post deployment verification steps, on page 38
- Automatic site failover, on page 39

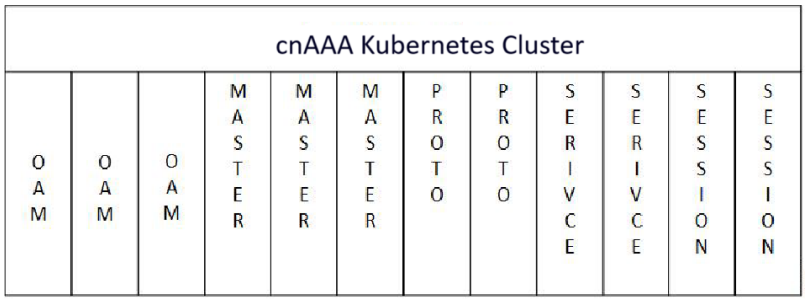
Introduction

Cisco cnAAA has a three-tier architecture which consists of Protocol, Service, and Session tiers. Each tier includes a set of microservices (pods) for a specific functionality. Within these tiers, there exists a Kubernetes Cluster comprising of Kubernetes (K8s) master, and worker nodes (including Operation and Management nodes).

For high availability and fault tolerance, a minimum of two K8s worker nodes are required for each tier. Multiple replicas can be assigned to each worker node. Kubernetes uses the StatefulSets controller to manage pods. The pods require a minimum of two replicas for fault tolerance.

The following figure depicts a cnAAA K8s Cluster – Master nodes, Operations, and Management (OAM) worker nodes, Protocol worker nodes, Service worker nodes, Session (data store) worker nodes.

Figure 3: cnAAA 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 cnAAA protocol-related pods for RADIUS Endpoint.
- Service worker nodes - These nodes host the cnAAA application-related pods that perform session management processing.
- Session worker nodes - These nodes host the database-related pods that store subscriber session data.

Base cnAAA Configuration

The cnAAA base configuration provides a comprehensive overview of the required configurations to make Policy Builder and cnAAA functional. This involves setting up the infrastructure needed for deploying cnAAA through the Subscriber Microservices Infrastructure (SMI) and configuring the Ops-Center to utilize cnAAA capabilities effectively over time.

The base configuration consists of the following steps

1. Deployment through SMI—All the network functions are deployed through the SMI platform. The platform simplifies the cloud-native NF deployments and monitors the NF performance while providing an integrated experience.
2. Configuring Ops Center—The cnAAA Ops Center provides an intuitive console for interacting with cnAAA in terms of configuring and gaining visibility into resources and features that you have subscribed to.

   The Ops Center lets you review the current and historical configurations corresponding to your environment. See Accessing the cnAAA Ops Center

Deploy and Validate cnAAA Software

This section describe different stages in deploying the cnAAA Software.

- Run the SMI Cluster Manager
- Deploy the cnAAA Software
- Validate the Deployment

Run the SMI Cluster Manager

The cnAAA software deploy or in-service procedure utilizes the K8s rolling strategy to update the pod images. In K8s rolling update strategy, the pods of a StatefulSet updates sequentially to ensure that the ongoing process remains unaffected. Initially, a rolling update on a StatefulSet causes a single pod instance to terminate. This process continues until all the replicas of the StatefulSet are updated. The terminating pods exit gracefully after completing all the ongoing processes.

Note Run the SMI sync operation for the cnAAA Ops Center and Cloud Native Common Execution Environment (CN-CEE).

For more information, see the SMI Cluster Manager - Deployment guide.

Deploy the cnAAA

This section describes the procedures involved in deploy cnAAA.

Stage the cnAAA Image

This section describes the procedure involved in staging the cnAAA image before initiating the deployment.

To stage the cnAAA image:

1. Download and verify the cnAAA image.
2. Log in to the SMI Cluster Manager node as an **ubuntu** user.
3. Copy the images to the **Uploads** directory.

   ```
   sudo mv <cpc_new_image.tar> /data/software/uploads
   ```

Note The SMI uses the new image present in the **Uploads** directory to upgrade.

4. Verify whether the image is retrieved up by the SMI for processing from the **Uploads** directory.

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

   Example:

   ```
   ubuntu@pocnAAA-cm01:~/temp_08072019_T1651$ sleep 30; ls /data/software/uploads
   ubuntu@pocpc-cm01:~/temp_08072019_T1651$
   ```

5. Verify whether the images were successfully picked up and processed.

   Example:

   ```
   auser@unknown:$ sudo du -sh /data/software/packages/*
   1.6G /data/software/packages/cee.2019.07
   5.3G /data/software/packages/cpc.2019.08-04
   16K /data/software/packages/sample
   ```

   The SMI unpacks the images into the **packages** directory successfully to complete the staging.

   For more information, refer the SMI Cluster Manager - Deployment

Note Similar steps can be followed to deploy ULB. For more information, refer the *ULB Configuration and Administration Guide*.

Trigger cnAAA

The cnAAA is triggered using the SMI cluster manager. To trigger cnAAA using SMI Cluster Manager, follow these configurations:

1. Log in to the SMI Cluster Manager console.
2. Run the following command to log in to the SMI Ops Center.

   ssh -p <port_number> admin@$(kubectl get svc -n smi | grep '.*netconf.*<port_number>' | awk '{ print $4 }')

   Example:

   ```
   ubuntu@pocnAAA-cm01:~$ ssh -p <port_number> admin@$(kubectl get svc -n smi | grep
   '.*netconf.*<port_number>' | awk '{ print $4 }')
   admin@<admin_ip_address> password: SMI-CONSOLE-PASSWORD
   Welcome to the CLI
   admin connected from <admin_ip_address> using ssh on
   ops-center-smi-cluster-manager-85869cf9b6-7j64k
   ```

3. Download the latest .tar from the URL.

   software-packages download *URL*

   Example:

   ```
   SMI Cluster Manager# software-packages download <URL>
   ```

   NOTES:

   - **software-packages download** *url*—Specify the software packages to be downloaded through HTTP/HTTPS.

4. Verify whether the TAR balls are loaded.

   software-packages list

   Example:

   ```
   SMI Cluster Manager# software-packages list
   [ cnAAA-2019-08-21 ]
   [ sample ]
   ```

   NOTES:

   - **software-packages list** —Specify the list of available software packages.

5. Update the product repository URL with the latest version of the product chart.

   config
   cluster *cluster_name*
   ops-centers *app_name cnAAA_instance_name*
   repository *url*
   exit
   exit

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# clusters test2
   SMI Cluster Manager(config-clusters-test2)# ops-centers cnAAA data
   SMI Cluster Manager(config-ops-centers-cnAAA/data)# repository <url>
   ```

```
SMI Cluster Manager(config-ops-centers-cnAAA/data)# exit
SMI Cluster Manager(config-clusters-test2)# exit
```

NOTES:

- **cluster** —Specify the K8s cluster.
- *cluster_name* —Specify the name of the cluster.
- **ops-centers** *app_name instance_name* —Specify the product Ops Center and instance. *app_name* is the application name. *instance_name* is the name of the instance.
- **repository** *url*—Specify the local registry URL for downloading the charts.

6. Run the **cluster sync** command to update to the latest version of the product chart. For more information on **cluster sync** command, see the Important section.

   clusters *cluster_name* **actions sync run**

   Example:

   ```
   SMI Cluster Manager# clusters test2 actions sync run
   ```



Important The cluster synchronization configure the cnAAA Ops Center, which in turn deploy the application pods (through **helm sync** command) one at a time automatically.

NOTES:

- **cluster** —Specify the K8s cluster.
- *cluster_name* —Specify the name of the cluster.
- **actions** —Specify the actions performed on the cluster.
- **sync run** —Triggers the cluster synchronization.

Configure Cluster Manager for Dual Stack Environment

Follow these steps to configure the Cluster Manager for a Dual Stack Environment.

Configure Unified API IPv6 support

Follow these steps to configure Unified API IPv6 support. It supports IPv6 for all management functionality, including Ingress httproutes, Grafana, PolicyBuilder, Netconf, and SSH. Dual-stack is supported, ensuring public and virtual IP addresses (VIPs) are accessible over both protocols.

Procedure

Step 1 Configure the cluster to operate in dual-stack mode and define IPv6 subnets for pods and services.

```
clusters cncps
  environment cncps-dev
    configuration ipv6-mode dual-stack
    configuration pod-subnet-ipv6 2001:DB8:1:1::/108
    configuration service-subnet-ipv6 2001:DB8:2:2::/108
```

```
  exit
  Configure node-specific IPv6 addresses for Secure Shell (SSH) and Kubernetes (k8s) within the
cluster definition.
  nodes cm
    k8s ssh-ipv6 2001:DB8:1::83
    k8s node-ipv6 2001:DB8:2::11
    os additional-ssh-ips [ 192.0.2.1 2001:DB8:1::83 ]
    initial-boot netplan vlans vlan2400
      addresses [ 192.0.2.11/24 2001:DB8:1:1::11/64 ]
    exit
    initial-boot netplan vlans vlan3594
      addresses [ 192.0.2.83/26 2001:DB8:1::83/64 ]
    exit
  exit
```

Step 2 Specify IPv6 addresses for Netconf, SSH, and Ingress httproutes hostnames for both the cluster manager and Ops-Center instances.

```
cluster-manager netconf-ipv6 2001:DB8:1::83
cluster-manager ssh-port 3022
cluster-manager httproutes-hostname 2001-db8-1--83.sslip.io
cluster-manager iso-download-ipv6 2001:DB8:1::83

ops-centers cee cee-cm
  netconf-ipv6 2001:DB8:1::83
  ssh-ipv6 2001:DB8:1::83
  httproutes-hostname 2001-db8-1--83.sslip.io
exit
```

Step 3 Configure OAM, virtual IP with IPv6 addresses.

```
virtual-ips oam
  vrrp-interface vlan3594
  vrrp-router-id 22
  ipv6-addresses 2001:DB8:1::88
    mask 64
    device vlan3594
  exit
  ipv6-addresses 2001:DB8:1::89
    mask 64
    device vlan3594
  exit
exit
```

Step 4 Bind the Ingress httproutes addon to both IPv4 and IPv6 addresses.

```
addons httproutes bind-ip-address 192.0.2.83
addons httproutes bind-ipv6-address 2001:DB8:1::83
addons httproutes bind-ip-address-internal 192.0.2.11
addons httproutes bind-ipv6-address-internal 2001:DB8:2::11
addons httproutes enabled
exit
```

Step 5 Enable IPv6 features within the engine configuration.

```
api unified engine-group production-rjio
  api unified externalIPs [ 192.0.2.87 2001:DB8:1::87 ]
  engine production-rjio
    properties com.broadhop.domain.ipv6.enable.feature
      value true
    exit
    properties com.broadhop.pep.ipv6.enable.feature
      value true
    exit
  policy-builder properties com.broadhop.pep.ipv6.enable.feature
```

```
    value true
  exit
exit
```

Step 6 Bind the Ingress httproutes addon to both IPv4 and IPv6 addresses.

```
addons ingress bind-ip-address 192.0.2.83
addons ingress bind-ipv6-address 2001:DB8:1::83
addons ingress bind-ip-address-internal 192.0.2.11
addons ingress bind-ipv6-address-internal 2001:DB8:2::11
addons ingress enabled
exit
```

```
addons httproutes bind-ip-address 192.0.2.83
addons httproutes bind-ipv6-address 2001:DB8:1::83
addons httproutes bind-ip-address-internal 192.0.2.11
addons httproutes bind-ipv6-address-internal 2001:DB8:2::11
addons httproutes enabled
exit
```

Step 7 Enable IPv6 features within the engine configuration.

```
api unified engine-group production-rjio
api unified externalIPs [ 192.0.2.87 2001:DB8:1::87 ]
engine production-rjio
properties com.broadhop.domain.ipv6.enable.feature
value true
exit
properties com.broadhop.pep.ipv6.enable.feature
value true
exit
policy-builder properties com.broadhop.pep.ipv6.enable.feature
value true
exit
exit
```

Monitor the Deployment of cnAAA

You can monitor the status of the deployment through SMI Cluster Manager Ops Center configurations:

```
config
  clusters cluster_name actions sync run upgrade-strategy concurrent debug
true
  clusters cluster_name actions sync logs
  monitor sync-logs cluster_name
   clusters cluster_name actions sync status
   end
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy concurrent debug true
SMI Cluster Manager# clusters test1 actions sync logs
SMI Cluster Manager# monitor sync-logs test1
SMI Cluster Manager# clusters test1 actions sync status
```

NOTES:

- **clusters** *cluster_name*—Specify the information about the nodes to be deployed. *cluster_name* is the name of the cluster.

- **actions**—Configures the actions performed on the cluster.
- **sync run**—Triggers the cluster synchronization.
- **sync logs**—Displays the current cluster synchronization logs.
- **sync status**—Displays the current status of the cluster synchronization.
- **debug true**—Enters the debug mode.
- **monitor sync logs** – Monitors the cluster synchronization process.

Important You can view the pod details after the upgrade through CEE Ops Center. For more information on pod details, see Viewing the Pod Details section.

Pod affinity relaxation for maximum resource utilization

Feature history

Feature Name	Release Information	Description
Strict anti-affinity is disabled, allowing multiple replicas per node for maximum resource utilization	2025.03.0	This feature provides configurable control over anti-affinity rules for engine pods in Kubernetes clusters. Disabling anti-affinity allows the scheduler to place multiple engine pod replicas on the same node, ensuring all replicas are deployed even in environments with limited node availability.

This feature enables configuration of engine pod scheduling behavior in Kubernetes clusters. When strict anti-affinity rules are enforced, only one engine pod replica is scheduled per node. In resource-constrained environments, it prevents deployment of all requested replicas. Setting the `scheduling-affinity` parameter to `false` strict anti-affinity is disabled, allowing multiple replicas per node and allowing multiple engine pod replicas on the same node. This ensures balanced pod deployment and flexibility across available nodes.

Scheduling affinity parameter

The scheduling-affinity parameter controls how engine pods are scheduled in the cluster.

- When set to `true` (default), strict anti-affinity is enforced. Only one engine pod replica is scheduled per node. If the number of requested replicas exceeds the number of available nodes, additional replicas remain in a pending state.
- When set to `false` , strict anti-affinity is disabled, allowing multiple replicas per node. The scheduler can place multiple engine pod replicas on the same node if sufficient resources are available. Pod placement is balanced automatically to prevent resource bottlenecks and maximize utilization.
- When the `scheduling-affinity` parameter is set to `false` , the `maxSkew` parameter is automatically set to 1. This configuration ensures that the difference in the number of engine pods assigned to any two

nodes does not exceed 1. This balanced distribution applies only when each node has enough memory to host additional pods.

Configure the scheduling-affinity parameter for an engine pod

Procedure

Step 1 Enter the Ops-Center configuration mode:

```
config
```

Step 2 Navigate to the Engine Pod Configuration:

```
engine <engine_name>
```

Step 3 Set the scheduling-affinity Parameter.

- By default, `scheduling-affinity` is set to `true` (strict anti-affinity)
- To allow multiple replicas per node, set:

  ```
  scheduling-affinity false
  ```

Step 4 Specify the Number of Replicas.

```
replicas <number_of_replicas>
```



Note Perform Step 3 and Step 4 only when the system is in a shut down state.

Step 5 Enter `commit` to save the changes and `exit` the configuration mode.

Example

Sample configuration:

```
engine cpc-green
 scheduling-affinity false
 replicas 8
exit
```



Note Ensure that the cluster has sufficient CPU and memory resources to support the specified number of replicas.

Disable the optional PODs through Ops Center configuration

Feature History

Feature Name	Release Information	Description
Disable the optional Pods through Ops-Center configuration	2025.03.0	This feature allows deactivation of certain optional pods in the cnAAA. This is done through Ops Center configuration. Disabling optional pods reduces resource consumption and simplifies deployments.

This feature allows disabling optional pods such as traceid, network-query, redis, and consolidated-aaa-logging. These pods deployed by default and consume system resources even when standard operations do not require them. Disabling a pod optimizes resource usage, simplifies management, and offers deployment flexibility, especially in resource-constrained environments.

Deactivate optional Pods through Ops-Center

Follow these steps to configure optional pods using the Ops CLI:

Procedure

Step 1 Enter Ops-Center configuration mode.

```
config
```

Step 2 To view currently disabled optional pods, enter the **show running-config pods-management disable-pods** command.

```
show running-config pods-management disable-pods
```

Sample output:

```
[m13-cnaaa/m13] cpc# show running-config pods-management disable-pods
Thu Oct  30 04:29:38.896 UTC+00:00
% No entries found.
[m13-cnaaa/m13] cpc#
```

Note If no pods are disabled, the output indicates "No entries found."

Step 3 Specify the pods which needs to be disable using the **pods-management disable-pods** command.

```
pods-management disable-pods <pod_name1> <pod_name2> ...
```

Sample configuration to disable pods:

```
cpc(config)# pods-management disable-pods
[ consolidated-aaa-logging  network-query  redis  traceid ]
Tue Jul  22 12:41:17.163 UTC+00:00
[m13-cnaaa/m13] cpc(config)# commit
```

Note To re-enable a pod, remove it from the disabled list. Use the **no** form of the command followed by the pod name(s).

```
no pods-management disable-pods <pod_name1> <pod_name2>
```

Sample configuration to re-enable multiple pods:

```
cpc(config)# no pods-management disable-pods [ consolidated-aaa-logging network-query traceid
 ]
Tue Jul  22 11:47:10.101 UTC+00:00
[m13-cnaaa/m13] cpc(config)# commit
Tue Jul  22 11:47:12.266 UTC+00:00
Commit complete.
[m13-cnaaa/m13] cpc(config)#
```

Step 4 Commit the configuration changes.

```
commit
```

After committing the changes and re-deploying or updating the system with the new configuration, the specified pods are either excluded (if disabled) or included (if enabled) in the Kubernetes.

Sample configuration file

This sample configuration file provided solely for your reference. Create and modify the configuration file according to the specific needs of the deployment.

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

```
software cnf cee-2025.01.1.i14
 url                            http://<Repo_Server_IP>/releases/cee/cee-2025.01.1.i14.tar
 user                           labuser
 password                       labuser
 accept-self-signed-certificate true
 sha256
a5d00f217d011a9c941592433f1254888fe50b0d55d0e73011913575f477ed02
exit
software cnf cpc.2026.01.0.x
 url                            http://<Repo_Server_IP>/releases/cpc/cpc.2026.01.0.x.tar
 user                           labuser
 password                       labuser
 accept-self-signed-certificate true
 sha256
aa871140c8ac914d7e74e0e3b109fe1d43fb9a44bc9c1c07d5fa6bae34150764
exit
software cnf ulb.2025.01.0.i6
 url      http://<Repo_Server_IP>/releases/ulb/ulb.2025.01.0.i6.tar
 user     labuser
 password labuser
 sha256   0efcdab729f9408053d5d3c7deb64e619d3919b045e78d3e3fb76c82ca18d373
exit
environments alpha-env
 ucs-server
```

```
exit
feature-gates alpha true
feature-gates test true
clusters alpha
 environment alpha-env
 configuration master-virtual-ip        <httproutes_Bind_IP>
 configuration master-virtual-ip-cidr   24
 configuration master-virtual-ip-interface vlan2400
 configuration additional-master-virtual-ip <Additional_VIP>
 configuration additional-master-virtual-ip-cidr 26
 configuration cni type cilium
 configuration additional-master-virtual-ip-interface vlan3594
 configuration virtual-ip-vrrp-router-id 21
 configuration enable-pod-security-policy false
 configuration ipv6-mode                dual-stack
 configuration pod-subnet               192.101.0.0/16
 configuration pod-subnet-ipv6          2002:192:101::/108
 configuration service-subnet           10.101.0.0/16
 configuration service-subnet-ipv6      2002:10:101::/108
 configuration size                     production
 configuration allow-insecure-registry  true
 configuration restrict-logging         false
 configuration cilium lb-algorithm random
 configuration cilium enable-legacy-host-routing true
 configuration cilium enable-bgp true
 configuration cilium enable-egress-gateway true
 node-defaults ssh-username cloud-user
 node-defaults ssh-connection-private-key
"REe0lk3jB+ck8NCRd1jt40XkE0G69lOOm3V9o0539m+IRWJuczXf5rid"
 node-defaults initial-boot default-user cloud-user
 node-defaults initial-boot default-user-ssh-public-key "ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAILqi81JXjSCWg18QMphgIBl2QiIUkur1uJS/kK1otSDq smi@cisco.com"
 node-defaults initial-boot default-user-password Csco@123
 node-defaults initial-boot default-user-password-expiration-days 999
 node-defaults initial-boot netplan ethernets eno5
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan ethernets eno6
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan ethernets ens1f0
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan ethernets ens1f1
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan ethernets ens9f0
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan ethernets ens9f1
  dhcp4 false
  dhcp6 false
 exit
 node-defaults initial-boot netplan bonds bd0
  dhcp4      false
  dhcp6      false
  optional   true
  interfaces [ eno5 eno6 ]
  parameters mode      active-backup
```

```
 parameters mii-monitor-interval 100
 parameters fail-over-mac-policy active
exit
node-defaults initial-boot netplan bonds bd1
 dhcp4      false
 dhcp6      false
 optional   true
 interfaces [ ens1f0 ens9f1 ]
 parameters mode      active-backup
 parameters mii-monitor-interval 100
 parameters fail-over-mac-policy active
exit
node-defaults initial-boot netplan bonds bd2
 dhcp4      false
 dhcp6      false
 optional   true
 interfaces [ ens1f1 ens9f0 ]
 parameters mode      active-backup
 parameters mii-monitor-interval 100
 parameters fail-over-mac-policy active
exit
node-defaults initial-boot netplan vlans vlan2400
 dhcp4 false
 dhcp6 false
 id    2400
 link  bd1
exit
node-defaults initial-boot netplan vlans vlan3594
 gateway4 10.84.117.65
 nameservers search [ mitg-bxb300.cisco.com ]
 nameservers addresses [ 10.84.96.130 ]
 id       3594
 link     bd0
exit
node-defaults k8s ssh-username cloud-user
node-defaults k8s ssh-connection-private-key "$8$5tiwVUlT9T"
node-defaults k8s max-pods 256
node-defaults ucs-server cimc bios configured-boot-mode Uefi
node-defaults ucs-server cimc bios uefi-secure-boot yes
node-defaults ucs-server cimc certificate rehydrate true
node-defaults os tuned enabled
node-defaults os tuned base-profile latency-performance
node-defaults os ntp enabled
node-defaults os ntp servers 10.192.1.11
exit
nodes master-1
 maintenance false
 k8s node-type       master
 k8s ssh-ip          10.192.1.22
 k8s ssh-ipv6        2002:10:192:1::22
 k8s sshd-bind-to-ssh-ip true
 k8s node-ip         10.192.1.22
 k8s node-ipv6       2002:10:192:1::22
 k8s node-labels smi.cisco.com/node-type oam
 exit
 k8s node-labels smi.cisco.com/node-type-2 protocol
 exit
 k8s node-labels smi.cisco.com/node-type-3 service
 exit
 k8s node-labels smi.cisco.com/node-type-4 session
 exit
 ucs-server cimc user admin
 ucs-server cimc password Csco@123
 ucs-server cimc storage-adaptor create-virtual-drive true
```

```
 ucs-server cimc remote-management sol enabled
 ucs-server cimc remote-management sol baud-rate 115200
 ucs-server cimc remote-management sol comport com0
 ucs-server cimc remote-management sol ssh-port 2400
 ucs-server cimc ip-address 10.84.117.76
 ucs-server cimc networking ntp enabled
 ucs-server cimc networking ntp servers 10.84.117.83
 exit
 initial-boot default-user cloud-user
 initial-boot default-user-password Csco@123
 initial-boot netplan vlans vlan2005
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.55.22/24 2002:20:50:55::22/64 ]
  id        2005
  link      bd2
 exit
 initial-boot netplan vlans vlan2008
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.58.22/24 2002:20:50:58::22/64 ]
  id        2008
  link      bd2
 exit
 initial-boot netplan vlans vlan2010
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.60.22/24 2002:20:50:60::22/64 ]
  id        2010
  link      bd2
 exit
 initial-boot netplan vlans vlan2400
  addresses [ 10.192.1.22/24 2002:10:192:1::22/64 ]
 exit
 initial-boot netplan vlans vlan3594
  addresses [ 10.84.117.84/26 ]
 exit
 os additional-ssh-ips [ 10.84.117.84 ]
exit
nodes master-2
 maintenance false
 k8s node-type       master
 k8s ssh-ip          10.192.1.23
 k8s ssh-ipv6        2002:10:192:1::23
 k8s sshd-bind-to-ssh-ip true
 k8s node-ip         10.192.1.23
 k8s node-ipv6       2002:10:192:1::23
 k8s node-labels smi.cisco.com/node-type oam
 exit
 k8s node-labels smi.cisco.com/node-type-2 protocol
 exit
 k8s node-labels smi.cisco.com/node-type-3 service
 exit
 k8s node-labels smi.cisco.com/node-type-4 session
 exit
 ucs-server cimc user admin
 ucs-server cimc password Csco@123
 ucs-server cimc storage-adaptor create-virtual-drive true
 ucs-server cimc remote-management sol enabled
 ucs-server cimc remote-management sol baud-rate 115200
 ucs-server cimc remote-management sol comport com0
 ucs-server cimc remote-management sol ssh-port 2400
 ucs-server cimc ip-address 10.84.117.77
 ucs-server cimc networking ntp enabled
```

```
 ucs-server cimc networking ntp servers 10.84.117.83
 exit
 initial-boot default-user cloud-user
 initial-boot default-user-password Csco@123
 initial-boot netplan vlans vlan2005
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.55.23/24 2002:20:50:55::23/64 ]
  id        2005
  link      bd2
 exit
 initial-boot netplan vlans vlan2008
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.58.23/24 2002:20:50:58::23/64 ]
  id        2008
  link      bd2
 exit
 initial-boot netplan vlans vlan2010
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.60.23/24 2002:20:50:60::23/64 ]
  id        2010
  link      bd2
 exit
 initial-boot netplan vlans vlan2400
  addresses [ 10.192.1.23/24 2002:10:192:1::23/64 ]
 exit
 initial-boot netplan vlans vlan3594
  addresses [ 10.84.117.85/26 ]
 exit
 os additional-ssh-ips [ 10.84.117.85 ]
exit
nodes master-3
 maintenance false
 k8s node-type       master
 k8s ssh-ip          10.192.1.24
 k8s ssh-ipv6        2002:10:192:1::24
 k8s sshd-bind-to-ssh-ip true
 k8s node-ip         10.192.1.24
 k8s node-ipv6       2002:10:192:1::24
 k8s node-labels smi.cisco.com/node-type oam
 exit
 k8s node-labels smi.cisco.com/node-type-2 protocol
 exit
 k8s node-labels smi.cisco.com/node-type-3 service
 exit
 k8s node-labels smi.cisco.com/node-type-4 session
 exit
 ucs-server cimc user admin
 ucs-server cimc password Csco@123
 ucs-server cimc storage-adaptor create-virtual-drive true
 ucs-server cimc remote-management sol enabled
 ucs-server cimc remote-management sol baud-rate 115200
 ucs-server cimc remote-management sol comport com0
 ucs-server cimc remote-management sol ssh-port 2400
 ucs-server cimc ip-address 10.84.117.78
 ucs-server cimc networking ntp enabled
 ucs-server cimc networking ntp servers 10.84.117.83
 exit
 initial-boot default-user cloud-user
 initial-boot default-user-password Csco@123
 initial-boot netplan vlans vlan2005
  dhcp4     false
```

```
  dhcp6     false
  addresses [ 20.50.55.24/24 2002:20:50:55::24/64 ]
  id        2005
  link      bd2
 exit
 initial-boot netplan vlans vlan2008
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.58.24/24 2002:20:50:58::24/64 ]
  id        2008
  link      bd2
 exit
 initial-boot netplan vlans vlan2010
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.60.24/24 2002:20:50:60::24/64 ]
  id        2010
  link      bd2
 exit
 initial-boot netplan vlans vlan2400
  addresses [ 10.192.1.24/24 2002:10:192:1::24/64 ]
 exit
 initial-boot netplan vlans vlan3594
  addresses [ 10.84.117.86/26 ]
 exit
 os additional-ssh-ips [ 10.84.117.86 ]
exit
nodes worker-1
 maintenance false
 k8s node-type worker
 k8s ssh-ip 10.192.1.25
 k8s ssh-ipv6 2002:10:192:1::25
 k8s node-ip 10.192.1.25
 k8s node-ipv6 2002:10:192:1::25
 k8s node-labels smi.cisco.com/node-type-2 protocol
 exit
 k8s node-labels smi.cisco.com/node-type-3 service
 exit
 k8s node-labels smi.cisco.com/node-type-4 session
 exit
 ucs-server cimc user admin
 ucs-server cimc password Csco@123
 ucs-server cimc storage-adaptor create-virtual-drive true
 ucs-server cimc remote-management sol enabled
 ucs-server cimc remote-management sol baud-rate 115200
 ucs-server cimc remote-management sol comport com0
 ucs-server cimc remote-management sol ssh-port 2400
 ucs-server cimc ip-address 10.84.117.79
 ucs-server cimc networking ntp enabled
 ucs-server cimc networking ntp servers 10.84.117.83
 exit
 initial-boot default-user cloud-user
 initial-boot default-user-password Csco@123
 initial-boot netplan vlans vlan2005
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.55.25/24 2002:20:50:55::25/64 ]
  id        2005
  link      bd2
 exit
 initial-boot netplan vlans vlan2008
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.58.25/24 2002:20:50:58::25/64 ]
```

```
  id        2008
  link      bd2
 exit
 initial-boot netplan vlans vlan2010
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.60.25/24 2002:20:50:60::25/64 ]
  id        2010
  link      bd2
 exit
 initial-boot netplan vlans vlan2400
  addresses [ 10.192.1.25/24 2002:10:192:1::25/64 ]
 exit
 initial-boot netplan vlans vlan3594
  addresses [ 10.84.117.81/26 ]
 exit
 os additional-ssh-ips [ 10.84.117.81 ]
exit
nodes worker-2
 maintenance false
 k8s node-type worker
 k8s ssh-ip 10.192.1.26
 k8s ssh-ipv6 2002:10:192:1::26
 k8s node-ip 10.192.1.26
 k8s node-ipv6 2002:10:192:1::26
 k8s node-labels smi.cisco.com/node-type-2 protocol
 exit
 k8s node-labels smi.cisco.com/node-type-3 service
 exit
 k8s node-labels smi.cisco.com/node-type-4 session
 exit
 ucs-server cimc user admin
 ucs-server cimc password Csco@123
 ucs-server cimc storage-adaptor create-virtual-drive true
 ucs-server cimc remote-management sol enabled
 ucs-server cimc remote-management sol baud-rate 115200
 ucs-server cimc remote-management sol comport com0
 ucs-server cimc remote-management sol ssh-port 2400
 ucs-server cimc ip-address 10.84.117.80
 ucs-server cimc networking ntp enabled
 ucs-server cimc networking ntp servers 10.84.117.83
 exit
 initial-boot default-user cloud-user
 initial-boot default-user-password Csco@123
 initial-boot netplan vlans vlan2005
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.55.26/24 2002:20:50:55::26/64 ]
  id        2005
  link      bd2
 exit
 initial-boot netplan vlans vlan2008
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.58.26/24 2002:20:50:58::26/64 ]
  id        2008
  link      bd2
 exit
 initial-boot netplan vlans vlan2010
  dhcp4     false
  dhcp6     false
  addresses [ 20.50.60.26/24 2002:20:50:60::26/64 ]
  id        2010
  link      bd2
```

```
 exit
 initial-boot netplan vlans vlan2400
  addresses [ 10.192.1.26/24 2002:10:192:1::26/64 ]
 exit
 initial-boot netplan vlans vlan3594
  addresses [ 10.84.117.115/26 ]
 exit
 os additional-ssh-ips [ 10.84.117.115 ]
exit
virtual-ips oam
 vrrp-interface vlan3594
 vrrp-router-id 22
 check-interface vlan2400
 exit
 check-interface vlan3594
 exit
 ipv4-addresses 10.84.117.88
  mask      26
  broadcast 10.84.117.127
  device    vlan3594
 exit
 ipv4-addresses <Ops_Center_VIP>
  mask      26
  broadcast 10.84.117.127
  device    vlan3594
 exit
 ipv4-addresses 10.192.1.27
  mask      24
  broadcast 10.192.1.255
  device    vlan2400
 exit
 hosts master-1
  priority      100
  preempt-delay 0
 exit
 hosts master-2
  priority      100
  preempt-delay 0
 exit
 hosts master-3
  priority      100
  preempt-delay 0
 exit
exit
virtual-ips v4bng
 vrrp-interface vlan2010
 vrrp-router-id 41
 check-interface vlan2400
 exit
 ipv4-addresses 20.50.60.200
  mask      24
  broadcast 20.50.60.255
  device    vlan2010
 exit
 hosts master-1
  priority 100
 exit
 hosts master-2
  priority 100
 exit
 hosts master-3
  priority 100
 exit
 hosts worker-1
```

```
  priority 100
 exit
 hosts worker-2
  priority 100
 exit
exit
virtual-ips v4ocs
 vrrp-interface vlan2005
 vrrp-router-id 42
 check-interface vlan2400
 exit
 ipv4-addresses 20.50.55.200
  mask      24
  broadcast 20.50.55.255
  device    vlan2005
 exit
 hosts master-1
  priority 100
 exit
 hosts master-2
  priority 100
 exit
 hosts master-3
  priority 100
 exit
 hosts worker-1
  priority 100
 exit
 hosts worker-2
  priority 100
 exit
exit
virtual-ips v6bng
 vrrp-interface vlan2010
 vrrp-router-id 61
 check-interface vlan2400
 exit
 ipv6-addresses 2002:20:50:60::200
  mask   64
  device vlan2010
 exit
 hosts master-1
  priority 100
 exit
 hosts master-2
  priority 100
 exit
 hosts master-3
  priority 100
 exit
 hosts worker-1
  priority 100
 exit
 hosts worker-2
  priority 100
 exit
exit
virtual-ips v6ocs
 vrrp-interface vlan2005
 vrrp-router-id 62
 check-interface vlan2400
 exit
 ipv6-addresses 2002:20:50:55::200
  mask   64
```

```
  device vlan2005
 exit
 hosts master-1
  priority 100
 exit
 hosts master-2
  priority 100
 exit
 hosts master-3
  priority 100
 exit
 hosts worker-1
  priority 100
 exit
 hosts worker-2
  priority 100
 exit
exit
ops-centers cee alpha
 repository-local        cee-2025.01.1.i14
 sync-default-repository true
 netconf-ip              <Ops_Center_VIP>
 netconf-port            2024
 ssh-ip                  <Ops_Center_VIP>
 ssh-port                22
 httproutes-hostname        <Additional_VIP>.nip.io
 initial-boot-parameters use-volume-claims true
 initial-boot-parameters first-boot-password Csco@123
 initial-boot-parameters auto-deploy true
 initial-boot-parameters single-node false
exit
ops-centers lbs alpha
 repository-local        ulb.2025.01.0.i6
 sync-default-repository true
 netconf-ip              10.84.117.88
 netconf-port            4024
 ssh-ip                  10.84.117.88
 ssh-port                22
 httproutes-hostname        <Additional_VIP>.nip.io
 initial-boot-parameters use-volume-claims false
 initial-boot-parameters first-boot-password Csco@123
 initial-boot-parameters auto-deploy true
 initial-boot-parameters single-node false
exit
ops-centers cpc alpha
 repository-local        cpc.2026.01.0.x <Repo_Server_IP>
 sync-default-repository true
 netconf-ip              10.84.117.88
 netconf-port            3024
 ssh-ip                  10.84.117.88
 ssh-port                22
 httproutes-hostname        <Additional_VIP>.nip.io
 initial-boot-parameters use-volume-claims true
 initial-boot-parameters first-boot-password Csco@123
 initial-boot-parameters auto-deploy false
 initial-boot-parameters single-node false
exit
addons istio enabled
addons cpu-partitioner enabled
addons cpu-partitioner tier small
addons httproutes bind-ip-address <Additional_VIP>
addons httproutes bind-ip-address-internal <Master_VIP>
addons httproutes enabled
exit
```

Access the cnAAA Ops-Center

This section describes how to access the cnAAA Ops-Center.

Access the cnAAA Ops-Center from the CLI console in the console application. You can access these from the master node:

1. **CLI:**

   ssh username@*ops_center_service_ip* **-p 2024**

cnAAA Health Check

You need to perform a health check to ensure that all the services are running and nodes are in ready state. To perform a health check:

1. Log in to master node and use the following configuration:

```
kubectl get pods -n smi
kubectl get nodes
kubectl get nodes --show-labels
kubectl get pod --all-namespaces -o wide
kubectl get pods -n cpc-wsp -o wide
kubectl get pods -n cee-wsp -o wide
kubectl get pods -n smi-vips -o wide
helm list
kubectl get pods -A | wc -l
```



Important Ensure that all the nodes are in the ready state before you proceed further. Use the kubectl get nodes command to display the node states.

Verify the Deployment

This section describes the procedures involved in verifying the deployment process.

View the Pod Details

View RADIUS pod details through the CEE Ops-Center.

To view RADIUS pod details, use this command in the CEE Ops-Center CLI:

cluster pods *instance_name pod_name* **detail**



Note

- **cluster pods**—Specify the current pods in the cluster.
- *instance_name*—Specify the name of the instance.
- *pod_name*—Specify the name of the pod.
- **detail**—Displays the details of the specified pod.

The following example displays the details of the pod named *alertmanager-0* in the *cnAAA-data* instance.

Example:

```
cee# cluster pods cnAAA-data alertmanager-0 detail
details apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    alermanager.io/scrape: "true"
    cni.projectcalico.org/podIP: "<ipv4address/subnet>"
    config-hash: "5532425ef5fd02add051cb759730047390b1bce51da862d13597dbb38dfbde86"
  creationTimestamp: "2020-02-26T06:09:13Z"
  generateName: "alertmanager-"
  labels:
    component: "alertmanager"
    controller-revision-hash: "alertmanager-67cdb95f8b"
    statefulset.kubernetes.io/pod-name: "alertmanager-0"
  name: "alertmanager-0"
  namespace: "cnAAA"
  ownerReferences:
  - apiVersion: "apps/v1"
    kind: "StatefulSet"
    blockOwnerDeletion: true
    controller: true
    name: "alertmanager"
    uid: "82a11da4-585e-11ea-bc06-0050569ca70e"
  resourceVersion: "1654031"
  selfLink: "/api/v1/namespaces/cnAAA/pods/alertmanager-0"
  uid: "82aee5d0-585e-11ea-bc06-0050569ca70e"
spec:
  containers:
  - args:
    - "/alertmanager/alertmanager"
    - "--config.file=/etc/alertmanager/alertmanager.yml"
    - "--storage.path=/alertmanager/data"
    - "--cluster.advertise-address=$(POD_IP):6783"
    env:
    - name: "POD_IP"
      valueFrom:
        fieldRef:
          apiVersion: "v1"
          fieldPath: "status.podIP"
    image: "<path_to_docker_image>"
    imagePullPolicy: "IfNotPresent"
    name: "alertmanager"
    ports:
    - containerPort: 9093
      name: "web"
      protocol: "TCP"
    resources: {}
    terminationMessagePath: "/dev/termination-log"
    terminationMessagePolicy: "File"
    volumeMounts:
    - mountPath: "/etc/alertmanager/"
      name: "alertmanager-config"
    - mountPath: "/alertmanager/data/"
      name: "alertmanager-store"
    - mountPath: "/var/run/secrets/kubernetes.io/serviceaccount"
      name: "default-token-kbjnx"
      readOnly: true
  dnsPolicy: "ClusterFirst"
  enableServiceLinks: true
  hostname: "alertmanager-0"
  nodeName: "for-smi-cdl-1b-worker94d84de255"
```

```
  priority: 0
  restartPolicy: "Always"
  schedulerName: "default-scheduler"
  securityContext:
    fsGroup: 0
    runAsUser: 0
  serviceAccount: "default"
  serviceAccountName: "default"
  subdomain: "alertmanager-service"
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: "NoExecute"
    key: "node-role.kubernetes.io/oam"
    operator: "Equal"
    value: "true"
  - effect: "NoExecute"
    key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    tolerationSeconds: 300
  - effect: "NoExecute"
    key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    tolerationSeconds: 300
  volumes:
  - configMap:
      defaultMode: 420
      name: "alertmanager"
    name: "alertmanager-config"
  - emptyDir: {}
    name: "alertmanager-store"
  - name: "default-token-kbjnx"
    secret:
      defaultMode: 420
      secretName: "default-token-kbjnx"
status:
  conditions:
  - lastTransitionTime: "2020-02-26T06:09:02Z"
    status: "True"
    type: "Initialized"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "Ready"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "ContainersReady"
  - lastTransitionTime: "2020-02-26T06:09:13Z"
    status: "True"
    type: "PodScheduled"
  containerStatuses:
  - containerID: "docker://821ed1a272d37e3b4c4c9c1ec69b671a3c3fe6eb4b42108edf44709b9c698ccd"

    image: "<path_to_docker_image>"
    imageID:
"docker-pullable:<path_to_docker_image>@sha256:c4bf05aa677a050fba9d86586b04383ca089bd784d2cb9e544b0d6b7ea899d9b"

    lastState: {}
    name: "alertmanager"
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: "2020-02-26T06:09:05Z"
  hostIP: "<host_ipv4address>"
  phase: "Running"
```

```
  podIP: "<pod_ipv4address>"
  qosClass: "BestEffort"
  startTime: "2020-02-26T06:09:02Z"
cee#
```

Verify the Helm Status

This section describes the procedure involved in verifying the helm status. You need to determine whether the deployed helm chart is listed in the helm list successfully.

To determine the helm status:

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

   Sample output:

```
[m13-cnaaa/m13] cpc# show helm charts version
Wed Oct  29 06:36:48.650 UTC+00:00
CHART                    INSTANCE                                VERSION
-------------------------------------------------------------------------------------
cpc-ops-center           cpc-m13-ops-center                      BUILD_2025.03.1.i54
cnat-cps-infrastructure  cpc-m13-cnat-cps-infrastructure         BUILD_2025.03.1.i54
cps-radius-ep            cpc-m13-cps-radius-ep                   BUILD_2025.03.1.i54
etcd-cluster             cpc-m13-etcd-cluster                    BUILD_2025.03.1.i54
network-query            cpc-m13-network-query                   BUILD_2025.03.1.i54
ngn-datastore            cpc-m13-ngn-datastore                   BUILD_2025.03.1.i54
cpc-config               cpc-m13-cpc-config                      BUILD_2025.03.1.i54
cpc-dashboard            cpc-m13-cpc-dashboard                   BUILD_2025.03.1.i54
cpc-engine-app           cpc-m13-cpc-engine-app-production-rjio  BUILD_2025.03.1.i54
cpc-oam-app              cpc-m13-cpc-oam-app                     BUILD_2025.03.1.i54
cpc-services             cpc-m13-cpc-services                    BUILD_2025.03.1.i54

[m13-cnaaa/m13] cpc# show helm charts status
Wed Oct  29 06:36:54.252 UTC+00:00
CHART                    INSTANCE                                STATUS
-------------------------------------------------------------------------------
cpc-ops-center           cpc-m13-ops-center                      deployed
cnat-cps-infrastructure  cpc-m13-cnat-cps-infrastructure         deployed
cps-radius-ep            cpc-m13-cps-radius-ep                   deployed
etcd-cluster             cpc-m13-etcd-cluster                    deployed
network-query            cpc-m13-network-query                   deployed
ngn-datastore            cpc-m13-ngn-datastore                   deployed
cpc-config               cpc-m13-cpc-config                      deployed
cpc-dashboard            cpc-m13-cpc-dashboard                   deployed
cpc-engine-app           cpc-m13-cpc-engine-app-production-rjio  deployed
cpc-oam-app              cpc-m13-cpc-oam-app                     deployed
cpc-services             cpc-m13-cpc-services                    deployed

[m13-cnaaa/m13] cpc#
```

Verify the Pods

This section describes the procedure involved in determining the pod and container status after upgrading cnAAA. You need to ensure that the pods and containers are up and running.

Use the following commands to view the cnAAA pod logs.

kubectl describe pod *pod_name* **-n** *namespace*

Note 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.

Centralized system information command for cnAAA

Feature history

Table 1: Feature history

Feature Name	Release Information	Description
Centralized system information command for cnAAA	2026.01.0	This feature introduces the `cpc-system about-system-info` CLI command to provide a comprehensive overview of the system's state such as component versions and externally accessible endpoint URLs. As a result, users no longer need to execute multiple commands to obtain system information.

This feature introduces the `cpc-system about-system-info` CLI command. This command provides a consolidated overview of the system state, including component versions and externally accessible endpoint URLs. It replaces the multiple kubectl and helm commands previously required to gather system information. When executed from the OpsCenter, the command displays the following details:

- **Component versions**:
  - **CPC version**: The overall version of the Cisco Policy Controller.
  - **CPC core component versions**: Detailed build information for individual CPC modules (for example, `pcf-cnat-cps-infrastructure`, `pcf-cps-diameter-ep-grouprx`, `pcf-ops-center`).
  - **Database version**: The version of MongoDB in use (for example, `MONGO VERSION v7.0.28`).
  - **CDL version**: The version of the Cisco Data Layer component.
  - **CNDP version**: The version of the SMI platform CNDP.
  - **ULB version**: The Unified Load Balancer version (potential addition).
  - **OS Version**: The underlying Operating System version of pod and node.

Endpoint URLs: he output lists Grafana, Policy Builder, CPC Central, Control Center, the CRD REST API, Unified API WSDL, Unified API single-SPR and multi-SPR endpoints, Unified API Proxy status, CPC/CEE/ULB Ops-Center APIs, Prometheus-hi-res API.

```
cpc-system about-system-info

################## ABOUT INFO #################

CPC VERSION

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

CPC VERSION                    BUILD_2026.02.0.i55

CPC CORE COMPONENT VERSION

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

pcf-beta-cncps-cnat-cps-infrastructure        BUILD_2026.02.0.i55
pcf-beta-cncps-cps-radius-ep                  BUILD_2026.02.0.i55
pcf-beta-cncps-etcd-cluster                   BUILD_2026.02.0.i55
pcf-beta-cncps-network-query                  BUILD_2026.02.0.i55
pcf-beta-cncps-ngn-datastore                  BUILD_2026.02.0.i55
pcf-beta-cncps-ops-center                     BUILD_2026.02.0.i55
pcf-beta-cncps-pcf-config                     BUILD_2026.02.0.i55
pcf-beta-cncps-pcf-dashboard                  BUILD_2026.02.0.i55
pcf-beta-cncps-pcf-engine-app-production-rjio BUILD_2026.02.0.i55
pcf-beta-cncps-pcf-oam-app                    BUILD_2026.02.0.i55
pcf-beta-cncps-pcf-services                   BUILD_2026.02.0.i55
pcf-beta-cncps-unified-api-proxy-ep           BUILD_2026.02.0.i55



CNDP VERSION

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


cee-beta-cncps-cee-ops-center  2026.02.1.i05



ULB VERSION

------------------------
```

```
lbs-beta-ops-center            BUILD_2025.04.0.i18



DATABASE VERSION

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


CDL VERSION                    2.2.0


MONGO VERSION                  v7.0.28



OS VERSION

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


POD OS VERSION                 Ubuntu 22.04.5 LTS

NODE OS VERSION                Ubuntu 24.04.4 LTS



CPC ENDPOINT URLs

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

Grafana                   https://grafana.10.84.117.93.nip.io
Policy Builder
https://pb.pcf-beta-cncps-pcf-engine-app-production-rjio.10.84.117.93.nip.io/pb
CPC Central
https://pb.pcf-beta-cncps-pcf-engine-app-production-rjio.10.84.117.93.nip.io
Control Center           https://pcf.controlcenter.10.84.117.93.nip.io
CRD REST API
https://crd-api.pcf-beta-cncps-pcf-engine-app-production-rjio.10.84.117.93.nip.io/custrefdata
Unified API WSDL        https://pcf.unified-api.10.84.117.91.nip.io/ua/wsdl/UnifiedApi.wsdl
Unified API (Single-SPR) https://pcf.unified-api.10.84.117.91.nip.io/ua/soap
Unified API (Multi-SPR)  https://pcf.unified-api.10.84.117.91.nip.io/apirouter
Unified API Proxy        Not installed
CPC Ops-Center API       https://restconf.pcf-beta-cncps-ops-center.10.84.117.93.nip.io
CEE Ops-Center API      https://restconf.cee-beta-cncps-cee-ops-center.10.84.117.93.nip.io
ULB Ops-Center API       https://restconf.lbs-beta-ops-center.10.84.117.93.nip.io
Prometheus-hi-res API    https://prometheus-hi-res.10.84.117.93.nip.io
TAC Manager              https://show-tac-manager.10.84.117.93.nip.io
Documentation
https://docs.cee-beta-cncps-cee-product-documentation.10.84.117.93.nip.io
```

Each Ops-Center has a dedicated `restconf` ingress that can be use to manage system configurations.

Sample configuration:

```
cloud-user@beta-master-2:~$ curl -X GET -H "Accept: application/yang-data+json" -H
"Content-Length: 0" -k
https://restconf.pcf-beta-cncps-ops-center.10.84.117.93.nip.io/restconf/data/radius/device-group=ASR9K/
 -u "admin:Csco@123"
{
  "cisco-mobile-policy-radius:device-group": [
    {
      "name": "ASR9K",
      "default-shared-secret": "$8$NN1TZKxYV0VhU/AwnyhDxXrbjefL6cYYq4rHW6Ah2Ks=",
      "default-coa-shared-secret": "$8$2RGVxv+EeM3dVkgfe+pKux7TWNSM6mdQlzG/LuDx7kU=",
      "coa-port": 3799,
      "coa-timeout-seconds": 3,
      "device": [
        {
          "name": "dev1",
          "ip": "30.50.60.104",
          "shared-secret": "$8$V7uy9wa05SgL9wfSZYg7qDvFvtDc2ERam5DkrPvJTCc=",
          "coa-shared-secret": "$8$OSBemFPAAkJaYYWlZEhQmlK4C6qgaszYjYaorvflo1w=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev2",
          "ip": "30.50.59.100",
          "shared-secret": "$8$jVN6+79cfNg3UcgbzfbOlY8zQlw1a3q/hIdbVrTravQ=",
          "coa-shared-secret": "$8$VejMqJxpPXZ1NOrtMYEcPYIT8C7VaApG45Myb6QrX3E=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev3",
          "ip": "30.50.53.100",
          "shared-secret": "$8$T6ImW7n7ivtyqbGT6AHB+sAjLm3pOyThgh51fsmEFO4=",
          "coa-shared-secret": "$8$tIhkvyd0hWV3Uctedg2ObOP89Hbg/1B1F6bOFm5b7gk=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev4",
          "ip": "30.50.54.100",
          "shared-secret": "$8$7d9Rvt2ErFWw1pEsRQL4NVXDD2Qr3cjS0jDvwi6z+t4=",
          "coa-shared-secret": "$8$G85seWs9cQqzDd16skDn8PMsZZIKprH2epNdWKAPHlI=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev5",
          "ip": "30.50.55.100",
          "shared-secret": "$8$3COUO14/oBtsfPtre7sT7A08ozc93m6KXfoD7kWwIjY=",
          "coa-shared-secret": "$8$8cwGZwRMB7N6pmvTfn/oh3PjnwR8No16j0SGd2Aw+Pc=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev6",
          "ip": "30.50.56.100",
          "shared-secret": "$8$C4ikcwpW9i359dr9KzdNcIzqYncuus8q7eLPEs3imYM=",
          "coa-shared-secret": "$8$zNRZ/G11/3OOKsphCOdfskP3Vg1NnH75u9omJtdbh2k=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev7",
          "ip": "30.50.57.100",
          "shared-secret": "$8$YGQInkFa9oi44QEmIl2sGc8PlZDtUeRJa2s4I0X5Djk=",
          "coa-shared-secret": "$8$k08oIbyuwvVrZURFzh4yWn712qFntEAK9c9MaPilVyI=",
          "loopback-addresses": ["12.3.1.2"]
```

```
        },
        {
          "name": "dev8",
          "ip": "30.50.58.100",
          "shared-secret": "$8$CJG3CUKZdBI0iebZO3gSoKkeDCfSfj48GJyxcrbSwVs=",
          "coa-shared-secret": "$8$r558D+wqGHOg0HbyVyhX8JeLRAFY7G6gyl1JQS86Ypg=",
          "loopback-addresses": ["12.3.1.2"]
        },
        {
          "name": "dev9",
          "ip": "30.50.60.100",
          "shared-secret": "$8$Up58AHsx6jnphvv10UrK1pU8gl6wlguCl9KsNm1odH4=",
          "coa-shared-secret": "$8$/EAmCsH7gHp6LaCmmvHMSr3+Pbg8j8iMTEgl0XPg26w=",
          "loopback-addresses": ["12.3.1.2"]
        }
      ]
    }
  ]
}
```

View the `unified-api` ingress details at the following location:

https://cisco.app.box.com/s/a3s9cw9qe9or81abd05i81j9m56j8txy/file/2035899361626

Use the crd-api ingress to retrieve the CRD table data.

```
curl --insecure https://crd-api.pcf-engine.192.0.2.1.example.com/custrefdata/Loopback/_query
```

Sample configuration:

```
curl --insecure
https://crd-api.pcf-beta-cncps-pcf-engine-app-production-rjio.10.84.117.93.nip.io/custrefdata/Loopback/_query
<rows>
    <row>
        <field code="Loopback" value=" Loopback1010111015"/>
        <field code="OLT_Name" value="match=INRJALWRSHTRTW6001ENBOLT001.*"/>
    </row>
    <row>
        <field code="Loopback" value="LOOPBK40TPS_31388"/>
        <field code="OLT_Name" value="match=INRJBHLRCDRSTW6001ENBOLT001.*"/>
    </row>
    <row>
        <field code="Loopback" value="LOOPBK40TPS_31389"/>
        <field code="OLT_Name" value="match=INRJGGGRJWHRNB0002ENBOLT001.*"/>
    </row>
    <row>
        <field code="Loopback" value="LOOPBK40TPS_31390"/>
        <field code="OLT_Name" value="match=INUWETWHSVEHNB0001NA2OLT001.*"/>
    </row>
</rows>
```

Known limitations

- **Software version**: Run the latest build version of the cnAAA system to access this feature. Older builds require an upgrade.
- Execution environment: Execute the command within the OpsCenter operational mode.
- **Accuracy of version information**: The accuracy of the displayed version information depends on correctly applied Docker image tags that follow a consistent pattern.

- **Endpoint discovery**: The script discovers only endpoints exposed through Kubernetes `Ingress` or `Service` resources. The script does not detect manually configured services or services exposed outside of Kubernetes.
- **Performance dependency**: The command's performance depends on the health and responsiveness of the Kubernetes API server. In a healthy cluster, execution time is minimal (a few seconds).

Check the PB, CRD, Ops-Center Configuration

Use Policy Builder, CRD, Grafana, Unified API, Control Center URLS to verify configuration files.

```
cloud-user@alpha-master-1:~$ kubectl get httproutes -n cpc-alpha
NAME CLASS HOSTS ADDRESS PORTS AGE
crd-api-httproutes-cpc-alpha-cpc-engine-app-production-rjio nginx
crd-api.cpc-alpha-cpc-engine-app-production-rjio.198.51.100.0/24.nip.io 192.0.2.0/24, 443
10h
patch-server-httproutes-cpc-alpha-cnat-cps-infrastructure nginx
patch.cpc-alpha-cnat-cps-infrastructure.203.0.113.0/24.nip.io 198.51.100.1, 443 10h
cpc-controlcenter-httproutes nginx cpc.controlcenter.192.0.2.254.nip.io 10.101.204.15 80,
443 10h
cpc-unified-api-httproutes nginx cpc.unified-api.<Additional_VIP>.nip.io 192.0.2.254, 443
10h
policy-builder-httproutes-cpc-alpha-cpc-engine-app-production-rjio nginx
pb.cpc-alpha-cpc-engine-app-production-rjio.<Additional_VIP>.nip.io 203.0.113.0/24, 80, 443
 10h
restconf-httproutes-cpc-alpha-ops-center nginx
restconf.cpc-alpha-ops-center.192.0.2.0/24.nip.io 192.0.2.0/24.80, 443 13h
cloud-user@alpha-master-1:~$ kubectl get httproutes -n cee-alpha|grep grafana
grafana-httproutes                          nginx   grafana.192.0.2.0/24.nip.io
                     203.0.113.0/24, 443   13h
cloud-user@alpha-master-1:~$
```

This section describes the procedure involved in verifying all the Policy Builder and CRD configuration files from the backup.

Verify and enable Policy Builder configuration

1. Log in to the master node as an **ubuntu** user.
2. Retrieve the CPC URL.

   Example:

   ```
   ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$ kubectl get httproutes -n $( kubectl get
    namespaces | grep -oP 'cnAAA-(\d+|\w+)' | cut -d\  -f1) | grep policy-builder | awk '{
    print $2 }'
   pb.cnAAA-02-cnAAA-engine-app-blv02.<ipv4address>.nip.io
   ```

3. Navigate to the CPC URL and log in with your user credentials.
4. Click **Import/Export** and click **Import**.
5. Click **File to Import**.
6. Choose the exported policy backed up in the **Back Up SVN, Policy, and CRD Data** section.
7. In **Import URL**tab, specify the following URL:

   `http://svn/repos/configuration`

8. Enter a brief description in **Commit Message** check box and click **Import**

Retrieve and use Cisco Policy Builder URL

1. Log in to the master node as an **ubuntu** user.
2. Run the following command to retrieve the Cisco Policy Builder URL.

   Example:

   ```
   kubectl get httproutes -n $(kubectl get namespaces | grep -oP 'cnAAA-(\d+|\w+)' | cut
   -d\  -f1) | grep policy-builder | awk '{ print "https://"$2"/pb" }'
   https://pb.cnAAA-02-cnAAA-engine-app-blv02.<ipv4address>.nip.io/pb
   ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$
   ```

3. Navigate to the Cisco Policy Builder URL and click the **Build Policies using version controlled data**
4. Choose **Repository** from the drop-down list. and click **OK**.
5. Log in with your user credentials.
6. Navigate to **File**and click **Publish to Runtime Environment**
7. Enter a brief description in **Commit Message** and click **OK**.

Verify and enable CRD data

1. In CPS Central home page, click **Custom Reference Data**.
2. Check the **Export CRD to Golden Repository** check box. Specify the SVN host name in **Please enter valid server Hostname or IP** field.

   For cnAAA the SVN host name value is *svn*.
3. Click **+** to add the specified host name .
4. Click **Export**.

Note You receive a success message when the data is exported successfully.

Verify and enable Ops-Center configuration

This section describes the procedure involved in enabling the cnAAA Ops-Center configuration after restoring it.

1. Log in to the master node as an **ubuntu** user.
2. Run the following command to log in to the cnAAA Ops-Center CLI.

   Example:

   ```
   ubuntu@pocnAAA-mas01:~$ ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get
   namespaces | grep -oP 'cnAAA-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }')
   admin@<admin_ip_address> password: cnAAA-OPS-PASSWORD
   Welcome to the cnAAA CLI on pocnAAA01
   admin connected from <admin_ip_address> using ssh on
   ops-center-cnAAA-01-ops-center-68dd9f588-htjdf
   ```

3. Paste the contents of the exported cnAAA configuration file (the **cpcops.txt** file) mentioned in the cnAAA Ops-Center.

   Example:

   ```
   product cnAAA# config
   Entering configuration mode terminal
   product cnAAA(config)# <PASTE CONTENTS OF cpcops.txt AND RETURN TO 'config' mode. Don't
    Paste Default Configuration>
   product cnAAA(config)#
   ```

Note Before adding or restoring a cnAAA backup configuration in cnAAA Ops-Center, all passwords must be in plain text format.

Important Fix any sections in the configuration file that did not import properly.

4. Ensure that the helm URLs are inline with the updated cnAAA image.

   Example:

   ```
   product cnAAA(config)# helm repository base-repos
   product cnAAA(config-repository-base-repos)# url <url>
   product cnAAA(config-repository-base-repos)# exit
   product cnAAA(config)# k8s registry <registry_url>
   product cnAAA(config)# commit
   Commit complete.
   product cnAAA(config)#
   ```

Note Before updating the Helm repository, check the status through this command. If the URL is correct with base-repos, then re-configuration in Ops-Center is not required.

```
[m13-cnaaa/m13] cpc# show running-config helm
Thu Oct  30 04:55:14.891 UTC+00:00
helm default-repository base-repos
helm repository base-repos
 url https://charts.2002-10-192-1--21.sslip.io/cpc.2025.03.1.i54
exit
[m13-cnaaa/m13] cpc#
```

Post deployment verification steps

Procedure

Step 1 Verify that the cnAAA software is running with the latest release.

```
cloud-user@m13-cnaaa-master-1:~$ helm ls -A| grep cpc
cpc-m13-cnat-cps-infrastructure         cpc-m13         2               2025-04-11 11:53:48.578067272
 +0000 UTC deployed        cnat-cps-infrastructure-0.6.10-dev-cpc-2025-02-0050-250408121948-f40a27d
    BUILD_2025.02.0.i64
cpc-m13-cps-radius-ep                   cpc-m13         2               2025-04-11 16:20:11.651559054
 +0000 UTC deployed        cps-radius-ep-0.6.43-dev-cpc-2025-02-0200-250408121849-8607cb3
    BUILD_2025.02.0.i64
cpc-m13-network-query                   cpc-m13         1               2025-04-11 11:52:11.315757946
 +0000 UTC deployed        network-query-0.5.4-dev-cpc-2025-02-0087-250408121740-e3b7c7c
    BUILD_2025.02.0.i64
cpc-m13-ops-center                      cpc-m13         14              2025-04-11 07:35:17.683951128
 +0000 UTC deployed        cpc-ops-center-0.6.32-dev-cpc-2025-02-0028-250408121815-e6c0a94
         BUILD_2025.02.0.i64
cpc-m13-cpc-dashboard                   cpc-m13         1               2025-04-11 11:52:11.317568845
 +0000 UTC deployed        cpc-dashboard-0.2.17-dev-cpc-2025-02-0180-250408121812-9158e52
    BUILD_2025.02.0.i64
cpc-m13-cpc-engine-app-production-rjio  cpc-m13         3               2025-04-11 16:21:16.676033778
 +0000 UTC deployed        cpc-engine-app-0.9.1-dev-cpc-2025-02-0757-250410073515-0989b9a
    BUILD_2025.02.0.i64
cpc-m13-cpc-oam-app                     cpc-m13         1               2025-04-11 11:52:11.341161764
 +0000 UTC deployed        cpc-oam-app-0.6.2-dev-cpc-2025-02-0021-250408121806-044882f
    BUILD_2025.02.0.i64
cpc-m13-cpc-services                    cpc-m13         1               2025-04-11 11:52:11.315586937
 +0000 UTC deployed        cpc-services-0.6.17-dev-cpc-2025-02-0081-250408121813-5ce3a4e
    BUILD_2025.02.0.i64
cloud-user@m13-cnaaa-master-1:~$
```

Step 2 SSH to the ops-center, enter "system mode running" in the configuration prompt, and then commit.

Step 3 Use the same commands as in step 1, and verify that all the pods and nodes are operational.

Step 4 Check the PB and CRD data is configured.

Step 5 Use the same commands as in step 1, and verify that all the pods and nodes are operational.

Automatic site failover

Automatic site failover

Automatic site failover is a configuration-gated webhook feature. It receives eligible critical alerts. It validates ownership and site-safety conditions. It isolates source-site external traffic. It verifies the result. It retains durable recovery state.

The feature provides these capabilities:

- The system operates three webhook receiver replicas: one pod is active, and two pods are in standby mode.
- It accepts alerts if they are firing and critical, and their names begin with the configured failover prefix.
- The cnAAA diagnostic wrapper runs, and the system requires a fresh, healthy remote-site record for automatic isolation.
- Per-occurrence ownership and a shared cross-site lock prevent duplicate or simultaneous actions.
- The system saves the configured RADIUS replica count and Unified API external IPs, and then changes the traffic flow.

- It sets the configured RADIUS replica count to zero, removes Unified API external IPs, and verifies the resulting state.
- It supports operator-controlled rollback and reset after recovery verification.

Note Automatic site failover does not shut down application pods or internal Services. Automatic traffic restoration and MongoDB/SCDB primary stepdown are outside the current feature scope.

system-failover command

Use the `system-failover` command hierarchy to configure automatic site failover in CPC and to display receiver status.

The feature is disabled by default. When enabled, CPC renders a three-replica webhook receiver. The four identity and credential settings are mandatory. Advanced tuning settings have defaults and do not normally require changes.

Command mode

Configuration commands: Global Configuration mode.

Status command: Exec mode.

Syntax

```
system-failover enable true|false
system-failover alert-name-prefix <prefix>
system-failover ops-center-user-name <username>
system-failover ops-center-password <password>
system-failover database-name <shared_scdb_replica_set_name>
system-failover remote-site-name <remote_site_name>
system-failover advance-tuning health-monitor-interval-seconds <seconds>
system-failover advance-tuning startup-grace-seconds <seconds>
system-failover advance-tuning replicas <count>
system-failover advance-tuning action-lease-duration-seconds <seconds>
system-failover advance-tuning leader-lease-duration-seconds <seconds>
system-failover advance-tuning leader-retry-period-seconds <seconds>
system-failover status
```

Configuration commands

`system-failover enable`

Enables or disables the feature. The feature is disabled by default.

`alert-name-prefix`

Sets the prefix required for eligible alert names. The configuration example uses `auto-failover-`.

`ops-center-user-name` and `ops-center-password`

Set the remote Ops-Center username and password. The password uses a suppressed-echo encrypted leaf.

database-name

Sets the shared SCDB replica set name.

remote-site-name

Sets the configured remote site name.

advance-tuning

This command sets the receiver tuning values. The documented values include a health-monitor interval of 300 seconds, a startup grace period of 3600 seconds, three replicas, an action lease duration of 60 seconds, a leader lease duration of 30 seconds, and a leader retry period of 5 seconds.

Status command

Enter the command in Exec mode:

```
system-failover status
```

The status output reports the local gate, remote health, MongoDB state, and next required operator step.

Automatic site failover workflow

Automatic site failover occurs if the source site is reachable but cannot complete end-to-end AAA or API transactions. The current validated workflow isolates new external traffic. It also keeps application pods and internal services running.

Summary

The workflow uses these components:

- CEE alert rules evaluate the KPI condition, threshold, severity, and hold time.
- Alertmanager forwards firing and resolved notifications to the stable webhook Service.
- The Lease-active webhook pod evaluates eligibility and owns the action.
- The embedded cnAAA diagnostic wrapper publishes local health to shared SCDB.
- Shared SCDB stores site health, alert history, action state, recovery snapshots, rollback results, and the cross-site lock.
- The fixed action executor changes RADIUS replica state, removes Unified API external IPs, and verifies the result.

The workflow fails closed when a required state is missing, stale, unhealthy, unreachable, ambiguous, or not verified.

Workflow

The workflow progresses through these stages:

1. CEE evaluates an initial RADIUS or Policy Engine condition and sends a notification when the condition remains true for its configured hold time.
2. Alertmanager forwards the notification to the stable webhook Service. The Service routes the notification to the Lease-active receiver pod.

3. After the configured startup grace period, the receiver accepts only a firing, critical notification whose alert name begins with the configured prefix. The receiver also requires action ownership and an executable durable state.
4. The receiver identifies the occurrence from the alert name and start time. Repeated deliveries of the same occurrence are suppressed, and a matching resolved notification archives the occurrence.
5. The active receiver publishes local diagnostics and requires an existing, healthy, unexpired remote-site record. It then acquires the shared cross-site lock so that only one site can isolate traffic.
6. The receiver saves the current RADIUS replica count and Unified API external IPs. It sets the configured RADIUS replica count to zero, waits for the RADIUS endpoint pods to stop, removes the Unified API external IPs, and verifies both results.
7. The receiver records success only after verification. It retains the recovery snapshot and blocks another automatic action until operator-controlled recovery and reset are complete.
8. An operator uses the saved snapshot to restore the RADIUS replica count and Unified API external IPs, verifies the restored state, and runs the reset action when the recovery state is ready.

Result

The source site no longer advertises the verified external Unified API and RADIUS ingress, while application pods and internal Services remain running. The recovery state and action result remain available in shared SCDB.

What's next

Use the operator status and recovery procedure to reconcile the site, restore the saved traffic state, and reset the durable action gate.

Configure automatic site failover

Configure the CPC receiver and CEE notifications for alert-triggered automatic site failover.

Automatic site failover is disabled by default. When enabled, CPC renders a three-replica webhook receiver. The receiver isolates a site only after it receives an eligible alert and all required safety checks pass.



Warning To maintain consistent SOAP API traffic, monitor failover notifications and immediately perform the manual EMA site mapping when a failover alert fires.

Before you begin

- Ensure that CEE Prometheus and Alertmanager are available.
- Ensure that you can access a Kubernetes control-plane node or console application.
- Obtain CPC and CEE Ops-Center access and the required credentials.
- Ensure that the shared SCDB replica set is reachable from both sites.
- Ensure that the initial CEE rules can evaluate the RADIUS and Policy Engine metrics.
- Obtain the configured remote site name and required Ops-Center credentials.

SUMMARY STEPS

1. Log in to the CPC Ops-Center CLI from a Kubernetes control-plane node.
2. Enter CPC configuration mode.
3. Configure the system-failover feature and its required values.
4. Exit CPC configuration mode and display the operational status.
5. Connect to the CEE Ops-Center CLI in a separate session.
6. Enter CEE configuration mode.
7. Configure the stable Alertmanager receiver.
8. Configure the initial automatic-action alert rules.
9. Verify the reconciled receiver and notification path.

DETAILED STEPS

Procedure

Step 1 Log in to the CPC Ops-Center CLI from a Kubernetes control-plane node.

```
ssh <username>@<cpc_ops_center_pod_ip> -p 2024
```

Step 2 Enter CPC configuration mode.

```
config
```

Confirm that the prompt displays `pcf(config)#`.

Step 3 Configure the system-failover feature and its required values.

```
system-failover enable true
system-failover alert-name-prefix auto-failover-
system-failover ops-center-user-name <username>
system-failover ops-center-password <password>
system-failover database-name <shared_scdb_replica_set_name>
system-failover remote-site-name <remote_site_name>
system-failover advance-tuning health-monitor-interval-seconds 300
system-failover advance-tuning startup-grace-seconds 3600
system-failover advance-tuning replicas 3
system-failover advance-tuning action-lease-duration-seconds 60
system-failover advance-tuning leader-lease-duration-seconds 30
system-failover advance-tuning leader-retry-period-seconds 5
```

The four identity and credential values are mandatory when the feature is enabled. Advanced tuning values are set to default and typically do not require changes. Enter the password through the suppressed-echo encrypted leaf.

Step 4 Exit CPC configuration mode and display the operational status.

```
exit
system-failover status
```

Confirm that the output displays the local gate, remote health, MongoDB state, and next required operator step.

Step 5 Connect to the CEE Ops-Center CLI in a separate session.

```
ssh -p <cee_ssh_port> admin@<cee_ops_center_ip>
```

Step 6 Enter CEE configuration mode.

```
config
```

Confirm that the prompt displays `cee(config)#`.

Step 7 Configure the stable Alertmanager receiver.

```
prometheus alert-receivers target ext
url http://<webhook-service>.<pcf-namespace>.svc.cluster.local:8888/webhook
exit
```

Use the stable webhook Service URL and the `/webhook` path. Do not use a pod IP.

Step 8 Configure the initial automatic-action alert rules.

For each rule, configure the source-site KPI expression, hold time, critical severity, and an alert name that begins with `auto-failover-`. The initial alert names are:

- `auto-failover-radius-pod-unavailable`
- `auto-failover-radius-access-request-missing-terminal-response`
- `auto-failover-policy-engine-unavailable`

CEE evaluates the KPI threshold and hold time. The webhook does not recalculate those values.

Use the CEE rule configuration syntax:

```
config alerts rules group <alert_group_name> rule <rule_name> expression <promql_expression> duration
 <duration> severity critical type <alert_type> annotation <annotation_name> value <annotation_value>
 end
```

Step 9 Verify the reconciled receiver and notification path.

```
kubectl get pods -n <namespace> -l component=webhook-receiver \
  -L webhook-receiver.cisco.com/role -o wide
kubectl get lease -n <namespace>
kubectl logs -n <namespace> <active-webhook-pod> --since=30m
curl -s http://127.0.0.1:8888/metrics
```

Confirm that three replicas are available. Verify that one pod has the active role, and that the remaining pods are standbys. Check that the stable Service has the expected endpoint.

If the expected state is absent, verify that the feature is enabled, the stable receiver URL is correct, and the startup grace period has elapsed. Ensure that the alert is firing, has critical severity, and uses the configured prefix.

CPC renders the configured system-failover receiver, CEE forwards notifications to the stable webhook Service, and the receiver can evaluate eligible alerts after the startup grace period.

What to do next

After an automatic action, use the operator recovery procedure to restore the saved traffic state and reset the durable action gate. A resolved alert or disabling the feature does not restore traffic.

Automatic site failover trigger scenarios

This topic lists the initial KPI conditions, eligibility checks, and verified source-site isolation results for automatic site failover. Use this information to identify the conditions that qualify for automatic site failover and the checks required before source-site isolation.

Table 2: Initial automatic site failover scenarios

Scenario	Trigger condition	Duration and severity	KPI	Action
RADIUS endpoint pod availability	At least 50% of configured RADIUS endpoint pods are unavailable.	3 minutes; critical.	RADIUS pod availability.	Set the configured RADIUS replica count to zero, verify endpoint pods stop, remove Unified API external IPs, and verify external IP is absent.
RADIUS authentication response	At least 50% of RADIUS pods produce no authentication responses while inbound requests continue.	1 minute; critical.	Authentication responses and inbound RADIUS requests.	Set the configured RADIUS replica count to zero, verify endpoint pods stop, remove Unified API external IPs, and verify external IP is absent.
Policy Engine pod availability	At least 50% of production Policy Engine pods are unavailable.	1 minute; critical.	Policy Engine pod availability.	Set the configured RADIUS replica count to zero, verify endpoint pods stop, remove Unified API external IPs, and verify external IP is absent.

For an alert to qualify, the notification must be firing, have critical severity, and use an alert name that begins with the configured prefix. Additionally, the startup grace period must have elapsed, the receiver must be Lease-active, the occurrence Lease must be available, durable state must allow a new action, the remote-site record must be healthy and fresh, and the shared cross-site lock must be acquired.

Initial alert rules

Configure the alert groups, rule names, durations, severities, and alarm types in this table.

Table 3: Initial alert rules

Alert group	Rule name	Duration and severity	Alarm type
`policy-engine-availability`	`auto-failover-policy-engine-pods-unavailable`	1 minute; critical.	Processing Error Alarm.
`radius-access-response-failure`	`auto-failover-radius-pod-access-request-missing-terminal-response`	1 minute; critical.	Processing Error Alarm.
`radius-endpoint-availability`	`auto-failover-radius-endpoint-pods-unavailable`	3 minutes; critical.	Processing Error Alarm.

Initial PromQL expressions

Use the source-backed expressions for the corresponding alert rules.

Table 4: Initial PromQL expressions by rule

Rule name	Expression
auto-failover-policy-engine-pods-unavailable	(((sum(kube_deployment_spec_replicas{deployment="pcf-engine-pcf-gamma-cncps-pcf-engine-app-pcf01production"}) - sum(kube_deployment_status_replicas_available{deployment="pcf-engine-pcf-gamma-cncps-pcf-engine-app-pcf01production"})) /sum(kube_deployment_spec_replicas{deployment="pcf-engine-pcf-gamma-cncps-pcf-engine-app-pcf01production"})) >= 0.5)and (sum(kube_deployment_spec_replicas{deployment ="pcf-engine-pcf-gamma-cncps-pcf-engine-app-pcf01production"}) > 0)
auto-failover-radius-pod-access-request-missing-terminal-response	(((sum by (pod) (rate(inbound_request_total{message_type="AccessRequest"}[5m])) - sum by (pod) (rate(radius_responses_total{message_type=~"AccessAccept|AccessReject"}[5m]))) / on (pod) sum by (pod) (rate(inbound_request_total{message_type="AccessRequest"}[5m]))) >= 0.5) and on (pod) (sum by (pod) (rate(inbound_request_total{message_type="AccessRequest"}[5m])) > 0)
auto-failover-radius-endpoint-pods-unavailable	((kube_statefulset_replicas{statefulset="radius-ep"} - kube_statefulset_status_replicas_ready{statefulset="radius-ep"}) / kube_statefulset_replicas{statefulset="radius-ep"} >= 0.5) and (kube_statefulset_replicas{statefulset="radius-ep"} > 0)

Alert annotations

Use the SME-provided annotations for the initial alert rules.

Table 5: Initial alert annotations

Rule name	Summary	Description
auto-failover-policy-engine-pods-unavailable	At least 50% of production Policy Engine pods are unavailable.	At least 50% of configured engine replicas have remained unavailable for 1 minute.
auto-failover-radius-pod-access-request-missing-terminal-response	At least 50% of Access-Requests on a radius-ep pod are missing a terminal response.	This alert fires when a radius-ep pod receives Access-Request traffic and at least 50% of those requests do not receive a final Access-Accept or Access-Reject response for 1 minute.

Rule name	Summary	Description
`auto-failover-radius-endpoint-pods-unavailable`	At least 50% of the configured RADIUS endpoint pods are unavailable.	At least 50% of the radius-ep StatefulSet replicas have remained unavailable continuously for 3 minutes. Initiate source-site failover.

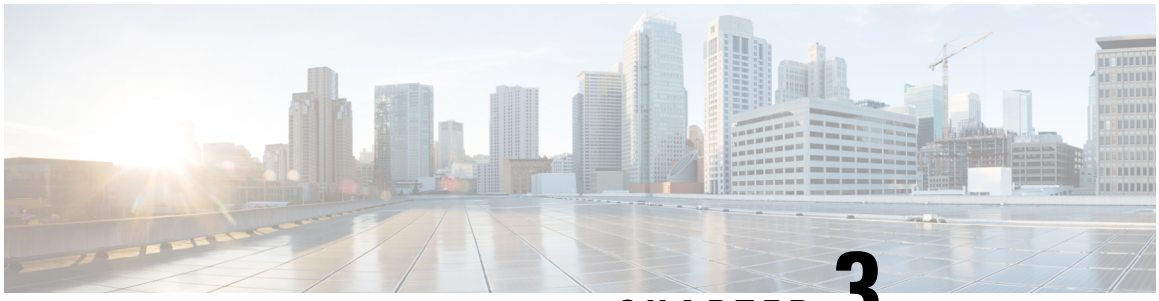
Note When a configured RADIUS-related alert condition triggers System Failover, the alert clears. Failover stops all RADIUS pods, so this is expected behavior.

Action outcomes

Table 6: Automatic site failover action outcomes

Category	Outcome
Alert-only or no action	The receiver acknowledges and logs resolved, non-critical, incorrectly named, startup-grace, duplicate, unowned, or safety-rejected notifications without changing external IP.
Source-site ingress isolation	After all mandatory checks pass, the receiver saves the recovery snapshot, sets the configured RADIUS replica count to zero, verifies the endpoint pods stop, removes Unified API external IP, verifies the result, records success, and sets durable state to completed.
Controlled recovery	An operator confirms source-site health, restores the saved RADIUS replica count and Unified API external IP, verifies both states, and resets the durable action gate.
Blocked or failed action	The receiver records the failure or ambiguous state, retains the blocked state, and requires operator reconciliation before another action is permitted.

Note If you are performing a cluster upgrade, disable this feature as pod restarts might occur during the upgrade process.



CHAPTER 3

User and Group Management

- Introduction, on page 49
- Add a user from Ops Center CLI, on page 49
- Delete a user from Ops Center CLI, on page 50
- Password change from Ops Center CLI, on page 50
- Update password length policy from Ops Center CLI, on page 51
- Add a user-group from Ops Center CLI, on page 51
- Delete a user-group from Ops Center CLI, on page 52
- Assign a user to a user-group, on page 52
- Unassign a user from a user-group, on page 53
- User privileges and access control in Ops-Center, on page 53

Introduction

The Ops Center CLI enables management of user accounts and groups. It allows to create, update, or delete users and set password policies and organize users into groups to control access. User and group management ensures that the appropriate access is assigned for Policy Builder, Control Center, and Ops Center.

Add a user from Ops Center CLI

Follow these steps to add a user from Ops Center CLI:

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to create a user:

```
smiuser add-user username <username> password <password>
```

Example:

```
pcf# smiuser add-user username user1 password Cisco@123
Wed Jul  2  09:26:45.907 UTC+00:00
message User user1 added
pcf#
```

Note Ensure that the password meets parameters:

- At least eight characters.
- Contains at least one lowercase letter, one uppercase letter, one digit, and one special character.
- Allowed special characters: ~, @, #, %, ^, &, *, (,), _, +, `, -, =, [,], :, ", ;, ', |, <, >, ?, ,, ., /, $.
- Do not include `{` or `}`.
- Do not start the password with `$`.
- Do not use simple or dictionary-based passwords.
- Avoid reusing previous passwords.

Delete a user from Ops Center CLI

Follow these steps to delete a user.

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to delete a user account in Ops Center CLI.

```
smiuser delete-user username <username>
```

Example:

```
pcf# smiuser delete-user username user1
Wed Jul  2  09:31:05.100 UTC+00:00
message User user1 deleted (No cleanup necessary)
pcf#
```

Password change from Ops Center CLI

Follow these steps to change the password from Ops Center CLI:

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to change the password:

```
smiuser change-self-password current_password <current_password>
new_password <new_password> confirm_password <new_password>
 [password_expire_days <number_of_days>]
```

Note Replace <*number_of_days*> with the number of days until the password expires (optional, default is 180 days).

Example:

To change the password from `Cisco@123` to `Cisco@345` with a 180-day expiration:

```
pcf# smiuser change-self-password current_password Cisco@123 new_password Cisco@345 confirm_password
 Cisco@345 password_expire_days 180
message Password updated successfully
```

To use the default expiration:

```
pcf# smiuser change-self-password current_password Cisco@123 new_password Cisco@345 confirm_password
 Cisco@345
message Password updated successfully
```

Update password length policy from Ops Center CLI

Follow these steps to update the minimum password length:

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to update length for user passwords:

```
smiuser update-password-length length <number_of_characters>
```

Example:

To set the minimum length to ten characters:

```
pcf# smiuser update-password-length length 10
message Password updated successfully
```

Add a user-group from Ops Center CLI

Follow these steps to add a user-group:

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to add a new user-group:

```
smiuser add-group groupname <group_name>
```

Example:

```
pcf# smiuser add-group groupname group1
Wed Jul  2  09:46:38.319 UTC+00:00
message Group group1 created successfully
pcf#
```

Delete a user-group from Ops Center CLI

Follow these steps to delete a user-group:

Procedure

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to delete a user-group:

```
smiuser delete-group groupname <group_name>
```

Example:

```
pcf# smiuser delete-group groupname group1
Wed Jul  2  09:48:48.164 UTC+00:00
message Group group1 deleted successfully
pcf#
```

Assign a user to a user-group

Procedure

Follow these steps to assign a user to a user-group:

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to assign a user to a user-group:

```
smiuser assign-user-group username <username> group <group_name>
```

Example:

To assign user1 to group1:

```
pcf# smiuser assign-user-group username user1 group group1
Wed Jul  2  09:50:42.624 UTC+00:00
message User user1 assigned to group successfully
pcf#
```

Unassign a user from a user-group

Procedure

Follow these steps to unassign a user from a user-group:

Step 1 Login to Ops Center CLI and enter the `config` mode.

Step 2 Enter this command to unassign a user from a group:

```
smiuser unassign-user-group username <username> group <group_name>
```

Example:

To unassign user1 to group1:

```
pcf# smiuser unassign-user-group username user1 group group1
Wed Jul  2  09:52:55.602 UTC+00:00
message User un-assigned from group group1 successfully
pcf#
```

User privileges and access control in Ops-Center

The user access to Policy-builder, CPC Ops-center, and Control-Center is determined by group membership.

This table lists access privileges for common group assignments:

Table 7: Access privileges for CEE-user-groups

Group	User	Grafana	PB	Ops-Center CPC	Ops-Center CEE	Control Center
Grafana-editor	user	yes(able to create dashboards and view)	no	no	yes(read only)	no
Grafana-admin	user	yes(RW)	no	no	yes(read only)	no

sadmin	user	no	no	no	yes(read only)	no
policy-admin	user	no	no	no	yes(read only)	no
admin	user	no	no	no	yes(RW)	no
Grafana-viewer	user	yes(RO)	no	no	yes(RO)	no
No group assigned	user	no	no	no	yes(RO)	no

Table 8: Access privileges for CPC user-groups

CPC user-group	User	Grafana	PB	Ops -Center CPC	Ops- Center CEE	Control Center
sadmin	user	no	yes-RO	yes-RO	no	no
policy-admin	user	no	yes-RO	yes-RO	no	no
admin	user	no	yes-RW	yes-RW	no	yes-RW
No group assigned	user	no	yes-RO	yes-RO	no	no
readonly	user	no	yes-RO	yes-RO	no	yes-RO

Predefined Groups:

- Default user groups are created during system installation. These groups have read-only access to Policy Builder and Ops-Center, but do not have access to Control Center.

User-Defined Groups:

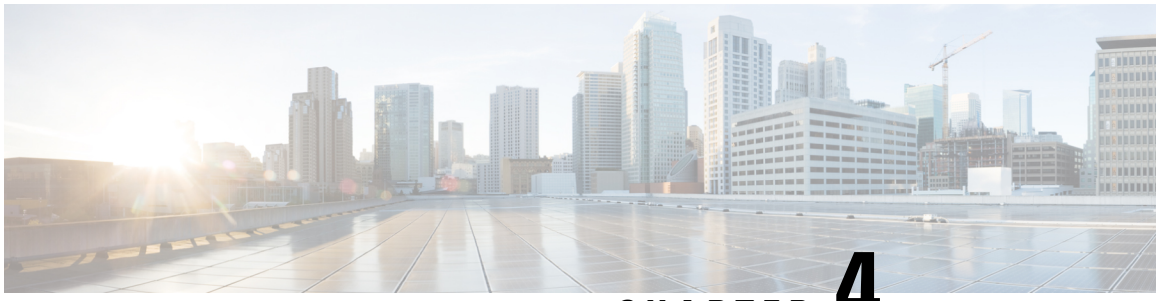
- Allows to create groups such as `group1` with custom privileges. By default, these groups provide Read-only access.

No Group Assigned:

- If a user is created but not assigned to any group, that user will have read-only access to Policy-Builder and Ops-Center, and no access to Control Center.

Security Principle:

- Assign users to the group with the minimum required privileges such as Read-only.



CHAPTER 4

Cisco Common Data Layer

- Feature History, on page 55
- Feature Description, on page 55
- How the CDL Works, on page 56
- Configure CDL through Ops Center, on page 60
- Configure the CDL Endpoints, on page 64

Feature History

Table 9: Feature History

Feature Details	Release
First Introduced	2025.01.0

Feature Description

The cnAAA extends support to the Geographic Redundancy (GR) version of the Cisco Common Data Layer (CDL). When the primary CDL endpoint fails, cnAAA attempts the same operation on the next highly rated secondary endpoint thus providing RADIUS message handling. If the next rated endpoint is unavailable, then cnAAA 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

This GR support feature has the following limitations:

- The cnAAA tries to reroute calls only when it encounters gRPC errors like "UNAVAILABLE." It ignores errors returned by the datastore and actual gRPC timeouts, such as the "DEADLINE_EXCEEDED" status code.
- The cnAAA Engine does not handle datastore failures, including indexing and slot failures. The CDL layer is responsible for resolving these issues and, if needed, making an API call to the remote system.

How the CDL Works

This section describes how this feature works.

Configure and Manage Endpoint Failover in CDL with cnAAA

Upon configuring the Cisco Common Data Layer (CDL) in cnAAA through the cnAAA Ops Center, the system supports multiple CDL datastore endpoints. Configuration involves specifying IP addresses, port numbers, and assigning ratings to each endpoint. By default, cnAAA considers the local endpoint as the primary one, with the highest rating. CDL API operations are initially performed on this primary endpoint. If the primary is unavailable, cnAAA routes operations to the next highest-rated endpoint. The system continues to fail over to the next accessible secondary endpoints until all configured endpoints are exhausted. cnAAA does not reattempt a query on an endpoint if it is reachable but responds with an error or timeout.

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

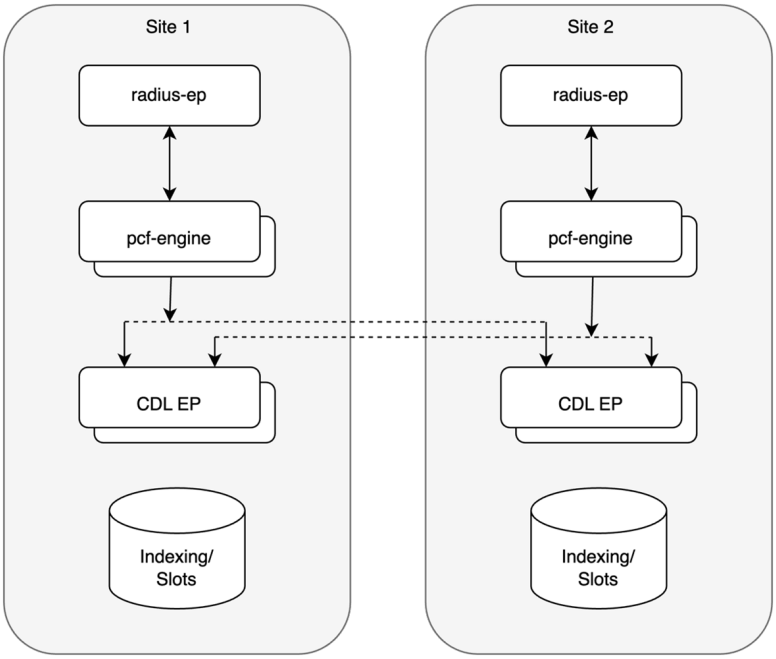
- cnAAA receives notification from CDL with session record from both the sites.
- After receiving the notification from CDL based on the session creation state only one site must processes the notification to resolve the conflict and save the session.

Architecture

You can configure CDL through cnAAA Ops Center. CDL in the GR mode replicates the session data across the configured sites. When cnAAA 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). cnAAA uses the secondary endpoints when the connection to the primary endpoint fails.

The following illustration depicts the failover that happens when the cnAAA Engine is unable to access the primary CDL datastore endpoint.

Figure 4: CDL Datastore Architecture



Call Flows

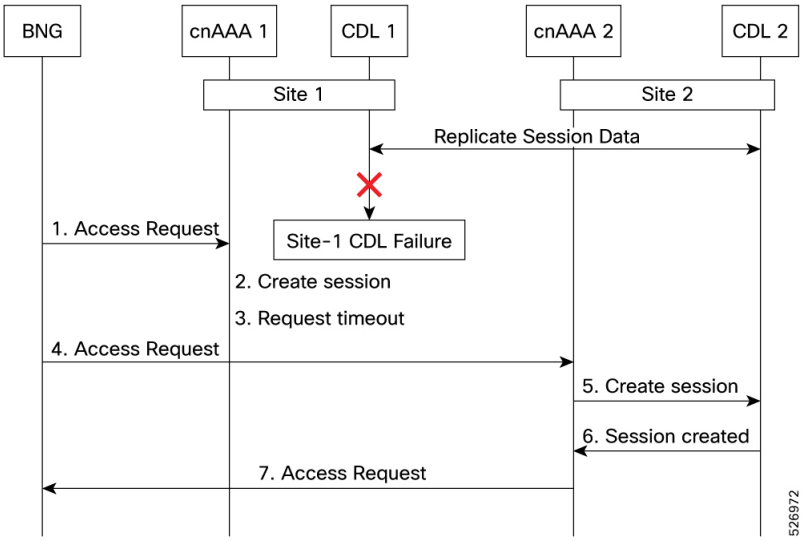
This section describes the key call flows for this feature.

CDL endpoint failure

CDL Endpoint Failure

A CDL endpoint failure occurs when the primary site's data layer becomes unreachable. The cnAAA relies on the CDL to manage session state. If the endpoint becomes unresponsive, the primary site cannot complete AAA transactions.

Figure 5: CDL endpoint failure call flow



These stages describe the call flow and system behavior during a CDL endpoint failure:

Stage	Description
1	The BNG sends a RADIUS Access-Request to Site 1.
2	Site 1 sends a session creation request to CDL 1.
3	Site 1 does not send a response because the CDL endpoint is down. The Access-Request times out on the BNG.
4	The BNG identifies Site 1 as unavailable and redirects the request to Site 2.
5	Site 2 sends a session creation request to CDL 2.
6	CDL 2 responds with a success message.
7	Site 2 sends a RADIUS Access-Accept message to the BNG.

GR Call Flows

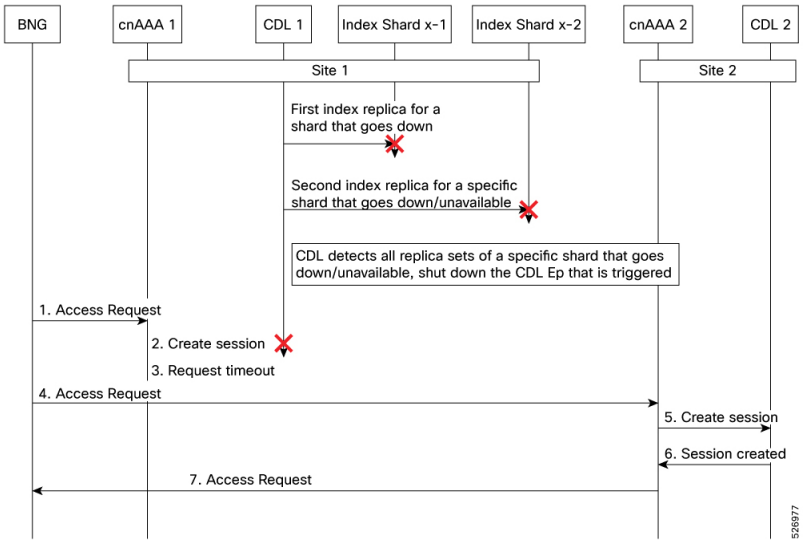
This section describes the possible CDL GR mode call flows scenarios that could start a failover to another site.

Indexing shard failure

An indexing shard failure occurs when two index replicas that belong to the same shard are unavailable. This scenario represents two points of failure, which typically occurs when replicas reside on different virtual machines or hosts.

If the primary CDL site (Site 1) is unavailable, the cnAAA RADIUS endpoint and engine redirect traffic to the secondary site (Site 2) based on the highest available rating.

Figure 6: Indexing shard failure call flow



These stages describe the sequence of events during an indexing shard failure:

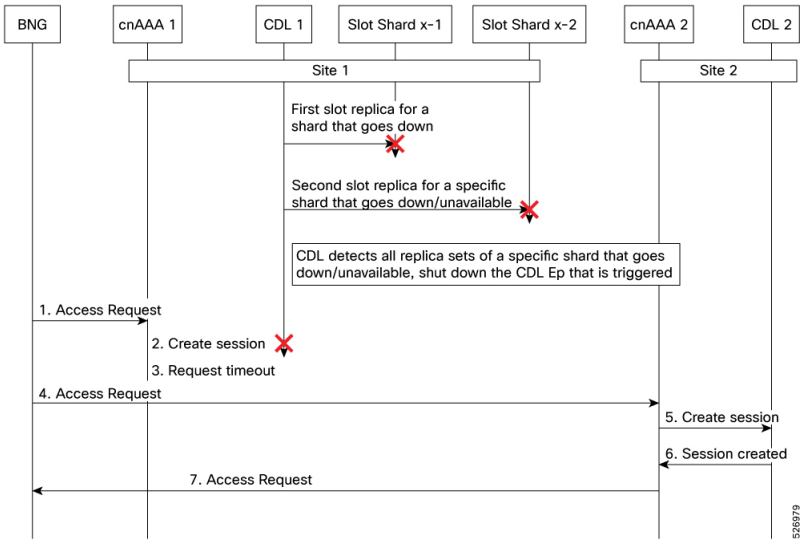
Table 10: Indexing shard failure call flow description

Stage	Description
1	The Broadband Network Gateway (BNG) sends a RADIUS Access-Request to Site 1.
2	The RADIUS endpoint receives the request and forwards it to the Policy Engine.
3	The Policy Engine attempts to send a session creation request to the CDL, but the connection fails.
4	Site 1 does not respond to the BNG, and the request times out.
5	The BNG identifies **Site 1** as unavailable and redirects the request to Site 2.
6	Site 2 sends a session creation request to CDL 2.
7	CDL 2 responds with a success message.

Slot replica set failure

A slot replica set failure occurs when two slot replicas that belong to the same replica set are unavailable. This scenario represents two points of failure, which typically occurs when replicas reside on different virtual machines or hosts.

Slot replica set failure call flow



These stages describe the sequence of events during a slot replica set failure:

Table 11: Slot replica set failure call flow description

Stage	Description
1	The BNG sends a RADIUS Access-Request to Site 1.
2	The RADIUS endpoint receives the request and forwards it to the Policy Engine.
3	The Policy Engine attempts to send a session creation request to the CDL, but the connection fails.
4	Site 1 does not respond to the BNG, and the request times out.
5	The BNG identifies Site 1 as unavailable and redirects the request to Site 2.
6	Site 2 sends a session creation request to CDL 2 and receives a success message.
7	Site 2 sends a RADIUS Access-Accept response to the BNG.

Configure CDL through Ops Center

This section describes how to configure the CDL endpoints.

Configure the CDL using cnAAA Ops Center involves the following 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 cnAAA.

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 the following configuration:

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

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

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
```

Configure the CDL Endpoints

This section describes how to configure the CDL endpoints.

Configure the CDL endpoints involves the following steps:

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

Configure the External Services

This section describes how to configure the external services in cnAAA.

CDL gets deployed in the GR environment as part of the SMI deployment procedure. By default, the CDL endpoints are available in the Datastore CLI node of the cnAAA Ops Center. However, you are required to configure these endpoints.

For each CDL site and instance, configure external service with the IP address and port number that corresponds to the site and instance.

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

```
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.

Associate the datastore with the CDL endpoint service

This section describes how to configure the external service for each CDL endpoint service that you plan to use.

To configure the external service for each CDL endpoint service, use the following configuration:

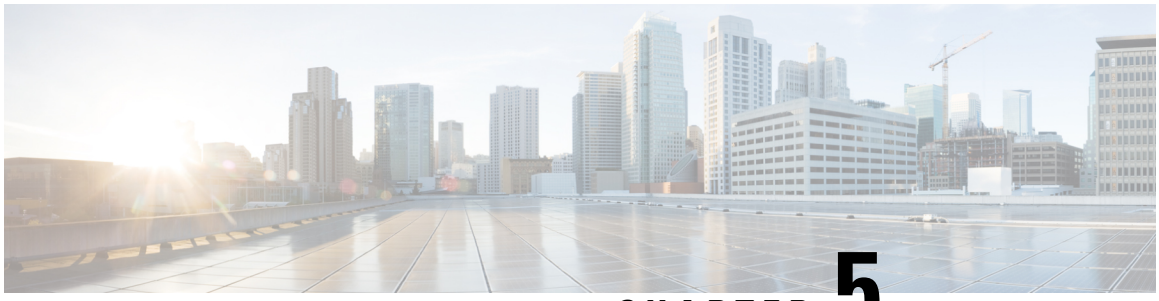
1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To associate the datastore with CDL endpoint service, use the following configuration:

```
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. cnAAA gives preference to the endpoints with the higher ratings.



CHAPTER 5

Software Upgrade Qualification

- Introduction, on page 67
- Deploy and validate cnAAA software, on page 68
- Update cnAAA Ops Center configuration, on page 82
- Restore the configuration from backup, on page 82

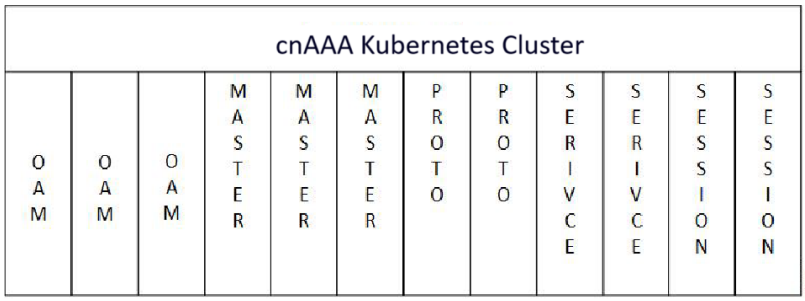
Introduction

Cisco cnAAA has a three-tier architecture which consists of Protocol, Service, and Session tiers. Each tier includes a set of microservices (pods) for a specific functionality. Within these tiers, there exists a Kubernetes Cluster comprising of Kubernetes (K8s) master, and worker nodes (including Operation and Management nodes).

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.

The following figure depicts a cnAAA K8s Cluster – Master nodes, Operations, and Management (OAM) worker nodes, Protocol worker nodes, Service worker nodes, Session (data store) worker nodes.

Figure 7: cnAAA 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 cnAAA protocol-related pods for RADIUS Endpoint.
- Service worker nodes - These nodes host the cnAAA application-related pods that perform session management processing.
- Session worker nodes - These nodes host the database-related pods that store subscriber session data.

Deploy and validate cnAAA software

This section describe different stages in deploying the cnAAA Software.

- Run the SMI Cluster Manager
- Deploy the cnAAA Software
- Validate the Deployment

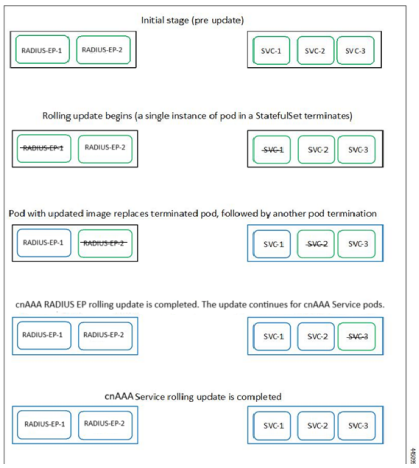
Run the SMI cluster manager

The cnAAA software deploy or in-service procedure utilizes the K8s rolling strategy to update the pod images. In K8s rolling update strategy, the pods of a StatefulSet updates sequentially to ensure that the ongoing process remains unaffected. Initially, a rolling update on a StatefulSet causes a single pod instance to terminate. This process continues until all the replicas of the StatefulSet are updated. The terminating pods exit gracefully after completing all the ongoing processes.

Note Run the SMI sync operation for the cnAAA Ops Center and Cloud Native Common Execution Environment (CN-CEE).

For more information, see the SMI Cluster Manager - Deployment guide.

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



Prerequisites

The prerequisites for upgrading cnAAA are:

- All the nodes – including all the pods in the node – are up and running.
- A patch version of the cnAAA software.

Note Currently, major versions do not support the rolling upgrade. The major version represents the release year, release number, and maintenance number. Cisco follows the versioning format as YYYY.RN.MN such as 2025.02.0.

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

cnAAA Health Check

Perform a health check to ensure that all the services are running and nodes are in ready state. To perform a health check:

1. Log in to master node.
2. Use these commands to view pod and node configurations:
   - View pods in the SMI namespace:

     ```
     kubectl get pods -n smi
     ```
   - View Kubernetes nodes:

     ```
     kubectl get nodes
     ```
   - View pods in all namespaces with wide output:

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

- View pods in the CPC namespace with wide output:

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

- View pods in the CEE namespace with wide output:

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

- List helm releases:

  ```
  helm list
  ```

- Count all pods across all namespaces:

  ```
  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 cnAAA Ops-Center with 100% functionality before starting traffic on it.



Important Ensure that all the nodes are in the ready state before you proceed further. Use the kubectl get nodes command to display the node states.

Check cnAAA Ops-Center Status

Use this command to view the status of the cnAAA Ops-Center:

- Use this command to view the system status:

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

Sample output:

```
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
```

Verify non-running Pods

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

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

Sample output:

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

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

Prepare for upgrade

This section describes the procedure involved creating a backup configuration, logs, and deployment files. To back up the files:

Table 12: 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 (e.g., `http://svn/repos//configuration`).
`username`	Username for authentication of PB or CRD.
`schedule-backup-daily`	Specifies the hour of the day when the backup operation should be executed (Values range from 00 to 23).
`schedule-backup-weekly`	Indicates the day of the week when the backup should occur (Values range from 1-7 and start with 1-Monday).

1. Log in to the SMI Cluster Manager Node as an **ubuntu** user.
2. Create a new directory for deployment.

   Example:

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

3. Move all the *cnAAA* deployment file into the newly created deployment directory.
4. Untar the *cnAAA* 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
```

5. Verify the downloaded image.

   Example:

   ```
   test@smi1cnAAA01-cm01:~/temp_08072019_T1651$ cat cpc.2025.01.SPA.tgZ
   ```

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

Backup SVN, Policy, and CRD data

This section describes the procedure involved in creating a backup of SVN, Policy, and CRD data. To perform a backup of SVN and Policy files:

1. Log in to the master node as an **ubuntu** user.
2. Use the following command to retrieve the Policy Builder URL.

   ```
   kubectl get ing -n $( kubectl get namespaces | grep -oP
   'cnAAA-(\d+|\w+)' | cut -d\  -f1) | grep policy-builder | awk '{ print
    $2 }'
   pb.cnAAA-02-cnAAA-engine-app-blv02.ipv4address.nip.io
   ```

   Example:

   ```
   cloud-user@m13-cnaaa-master-2:~$ kubectl get ing -n $( kubectl get namespaces | grep -oP
    'pcf-(\d+|\w+)') |grep policy-builder | awk '{ print $3 }'
   pb.pcf-m13-pcf-engine-app-production-rjio.10.84.16.218.nip.io
   cloud-user@m13-cnaaa-master-2:~$
   ```

   Sample output:

   ```
   pb.cnAAA-02-cnAAA-engine-app-blv02.ipv4address.nip.io
   ```

3. Navigate to the Policy Builder home page.
4. Click **Import/Export**.
5. Click **All Data**.
   - **Export URL**—Specify the export URL.
   - **Export File Prefix**—Specify an appropriate name for the export file.
6. Click **Export**.

Important You can find the exported file in your local **Downloads** directory.

To perform a backup of CRD data:

1. Navigate to the Policy Builder Home page.

2. Click **Custom Reference Data.**
3. Click **Import/Export CRD data.**
4. Click **Export**.

Important You can find the CRD data in your Web browser's **Downloads** directory.

Auto-backup of PB,CRD and Ops Center configurations

Feature History

Feature Name	Release Information	Description
Auto-backup of PB,CRD and Ops Center configurations	2025.03.0	This feature allows for automated backups of PB, CRD, and Ops-center configurations through customizable schedules and on-demand triggers. It consolidates backup processes, replacing manual tasks with automation, thereby enhancing operational efficiency.

Overview

A new Ops Center command-line utility simplifies the backup of critical configurations within the CPC system. This utility provides single-command backups by consolidating Ops Center, Policy Builder, and Customer Reference Data backups, which previously required separate API commands to trigger the backup for individual configurations. It also supports remote storage through Secure Copy Protocol (SCP) and allows scheduling of automated backups for better management and avoiding data loss. The utility improves backup management, helps prevent data loss, and enables restoring critical configuration.

Auto-backup procedure

Follow these steps to configure the backup settings from Ops Center:

Procedure

Step 1 Enter to `config` mode and configure parameters under `debug backup-config`.

This table lists the mandatory options for backup-config:

Option	Description
`backup-type`	Specifies the type of backup to perform (all, PB, CRD, Ops-center).

Option	Description
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 (e.g., `http://svn/repos//configuration`).
username	Username for authentication of PB or CRD.
schedule-backup-daily	Specifies the hour of the day when the backup operation should be executed (Values range from 00 to 23).
schedule-backup-weekly	Indicates the day of the week when the backup should occur (Values range from 1-7 and start with 1-Monday).

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

Example:

This sample configuration shows backup settings for the Ops Center:

```
[unknown] pcf# show running-config debug backup-config
debug backup-config backup-type all
debug backup-config username admin
debug backup-config password $8$pjU5UYsnL9kn5Lg5A1ckCB0fZx6qL5/fLWxNSMCk4/4=
debug backup-config svn-url http://svn/repos//configuration
debug backup-config scp-server-user-name cloud-user
debug backup-config scp-server-user-ip 10.84.117.99
debug backup-config scp-server-dest-backup-path /home/cloud-user/aellendu
debug backup-config scp-server-user-password $8$mEE8Kc5kW+5bIf4vYK1zpTaHYqUhYCUT05snEYl8c4Y=
```

Example

Note To trigger a backup manually, execute this command from Ops Center CLI:

pcf# `pcf-system periodic-backup backup-type all http://svn/repos/new_import123456`

- **backup-type all**: Initiates a backup of PB, CRD and Ops Center configurations.

Manual backup CEE and cnAAA Ops-Center configuration

This section describes the procedure involved in creating a backup of CEE and Ops Center configuration from the master node.

Procedure

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

Step 2 Create a directory to backup the configuration files.

```
mkdir backups_$(date +'%m%d%Y_T%H%M') && cd "$_"
```

Step 3 Back up the cnAAA Ops-Center configuration and verify the line count of the backup files.

```
ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces|
 grep -oP 'cnAAA-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }') "show run |
nomore" > cnAAAops.backup_$(date +'%m%d%Y_T%H%M') && wc-l cnAAAops.backup_$(date +'%m%d%Y_T%H%M')
```

Example:

```
ssh -p 2024 admin@$(kubectl get svc -n $(kubectl get namespaces| grep -oP 'pcf-(\d+|\w+)')
| grep 2024 | awk '{ print $3}') "show run | nomore" > cnAAAops.backup_$(date
+'%m%d%Y_T%H%M') && wc -l cnAAAops.backup_$(date +'%m%d%Y_T%H%M')
```

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

```
ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces
| grep -oP 'cee-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }') "show run | nomore" >
ceeops.backup_$(date +'%m%d%Y_T%H%M') && wc
-l ceeops.backup_$(date +'%m%d%Y_T%H%M')
```

Example:

```
ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$ ssh -p <port_number>
admin@$(kubectl get svc -n $(kubectl get namespaces | grep -oP 'cee-(\d+|\w+)') |
grep <port_number> | awk'{ print $3 }') "show run | nomore" > ceeops.backup_$
(date +'%m%d%Y_T%H%M') && wc -l ceeops.backup_$(date +'%m%d%Y_T%H%M')
admin@<admin_ip_address> password: CEE-OPS-PASSWORD 233 ceeops.backup
```

Step 5 Move the SMI Ops Center backup file (from the SMI Cluster Manager) to the backup directory.

```
scp $(grep cm01 /etc/hosts | awk '{ print $1
}'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M')
```

Example:

```
ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$ scp $(grep cm01 /etc/hosts | awk '{ print
$1 }'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M'). ubuntu@<admin_ip_address>
password: SMI-CM-PASSWORD smiops.backup 100% 9346 22.3MB/s 00:00
```

Step 6 Verify the line count of the backup files.

Example:

```
ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$ wc -l *
233 ceeops.backup
334 cnAAAops.backup
361 smiops.backup
928 total
```

Upgrade the cnAAA

This section describes the procedures involved in upgrading cnAAA.

• Staging a new cnAAA Image
• Triggering the Rolling Software Upgrade
• Monitoring the Upgrade

Stage a new cnAAA image

This section describes the procedure involved in staging a new cnAAA image before initiating the upgrade.

To stage the new cnAAA image:

1. Download and verify the new cnAAA image.

   ```
   url: http://<Repo_Server_IP>/releases/cpc/cpc.2026.01.0.x<Repo_Server_IP>.tar sha256:
   6817692f1703a657a1867b0ccaf4483fadd7a908d68094a7deb5d16ce234c0e7
   ```

2. Log in to the SMI Cluster Manager node as an **ubuntu** user.
3. Copy the images to **Uploads** directory.

   sudo mv <cnAAA_new_image.tar> /data/software/uploads



Note The SMI uses the new image present in the **Uploads** directory to upgrade.

4. Verify whether the image is picked up by the SMI for processing from the **Uploads** directory.

   sleep 30; ls /data/software/uploads

   Example:

   ```
   ubuntu@pocnAAA-cm01:~/temp_08072019_T1651$ sleep 30; ls /data/software/uploads
   ubuntu@pocnAAA-cm01:~/temp_08072019_T1651$
   ```

5. Verify whether the images were successfully picked up and processed.

   Example:

   ```
   auser@unknown:$ sudo du -sh /data/software/packages/*
   1.6G /data/software/packages/cee.2019.07
   5.3G /data/software/packages/cnAAA.2019.08-04
   16K /data/software/packages/sample
   ```

   The SMI unpacks the images into the **packages** directory successfully to complete the staging.

   For more information, refer the SMI Cluster Manager - Deployment



Note The SMI must unpack the images into the **packages** directory successfully to complete the staging.

Trigger the rolling software upgrade

The cnAAA is triggered using the SMI cluster manager. To trigger cnAAA using SMI cluster manager, follow these configurations:

1. Log in to the SMI cluster manager console.

2. Run the following command to log in to the SMI Ops Center.

   ssh -p <port_number> admin@$(kubectl get svc -n smi | grep '.*netconf.*<port_number>' | awk '{ print $4 }')

   Example:

   ```
   cloud-user@m13-cm:~$ ssh -p 2024 admin@$(kubectl get svc -n smi-cm | grep '2024' | awk
   '{ print $3 }')
   admin@10.101.154.93's password:

         Welcome to the Cisco SMI Cluster Deployer on m13-cm
         Copyright © 2016-2020, Cisco Systems, Inc.
         All rights reserved.

   admin connected from 10.192.1.11 using ssh on
   ops-center-smi-cluster-deployer-6685dd9dd9-52wk2
   [m13-cm] SMI Cluster Deployer#
   ```

3. Download the latest .tar from the URL.

   software-packages download *URL*

   Example:

   ```
   [m13-cm] SMI Cluster Deployer(config)# software cnf cpc.2025.02.0.i64
   [m13-cm] SMI Cluster Deployer(config-cnf-cpc.2025.02.0.i64)#  url
   http://10.84.16.209/releases/cpc/cpc.2025.02.0.i64.tar
   ```

   NOTES:

   - **software-packages download** *url*—Specify the software packages to be downloaded through HTTP/HTTPS.

4. Verify whether the TAR balls are loaded.

   software-packages list

   Example:

   ```
   [m13-cm] SMI Cluster Deployer# show running-config software cnf cpc.2025.02.0.i64
   ```

   NOTES:

   - **software-packages list** —Specify the list of available software packages.

5. Update the product repository URL with the latest version of the product chart.

   config
   cluster *cluster_name*
   ops-centers *app_name cnAAA_instance_name*
   repository-local *local repository name*
   exit
   exit

   Example:

   ```
   SMI Cluster Manager# config
   SMI Cluster Manager(config)# clusters m13-cnaaa
   SMI Cluster Manager(config-clusters-test2)# ops-centers pcf m13
   SMI Cluster Manager(config-ops-centers-cnAAA/data)# repository-local cpc.2025.02.0.i64
   SMI Cluster Manager(config-ops-centers-cnAAA/data)# exit
   SMI Cluster Manager(config-clusters-test2)# exit
   ```

NOTES:

- **cluster** —Specify the K8s cluster.
- *cluster_name* —Specify the name of the cluster.
- **ops-centers** *app_name instance_name* —Specify the product Ops Center and instance. *app_name* is the application name. *instance_name* is the name of the instance.
- **repository** *url*—Specify the local registry URL for downloading the charts.

6. Run the **cluster sync** command to update to the latest version of the product chart. For more information on **cluster sync** command, see the section.

   clusters *cluster_name* **actions sync run**

   Example:

```
clusters m13-cnaaa actions sync run sync-phase opscenter debug true   <<<<<<<<<<<<< (For
 Cluster Sync)
This will run sync.  Are you sure? [no,yes] yes
message accepted
warning "k8s node-type master" for node master-1 is deprecated. Use "k8s node-type
control-plane" instead
warning "k8s node-type master" for node master-2 is deprecated. Use "k8s node-type
control-plane" instead
warning "k8s node-type master" for node master-3 is deprecated. Use "k8s node-type
control-plane" instead
[m13-cm] SMI Cluster Deployer# monitor sync-logs m13-cnaaa     <<<<<<<<<<<<< (To monitor
 the sync logs)
```



Important The cluster synchronization configure the cnAAA Ops Center, which in turn upgrade the application pods (through **helm sync** command) one at a time automatically.

NOTES:

- **cluster** —Specify the K8s cluster.
- *cluster_name* —Specify the name of the cluster.
- **actions** —Specify the actions performed on the cluster.
- **sync run** —Triggers the cluster synchronization.

Monitor the cnAAA Upgrade

Monitor the status of the upgrade through SMI Cluster Manager Ops-Center configurations:

```
config
  clusters cluster_name actions sync run upgrade-strategy concurrent debug
true
  clusters cluster_name actions sync logs
  monitor sync-logs cluster_name
   clusters cluster_name actions sync status
   end
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run
SMI Cluster Manager# clusters test1 actions sync run upgrade-strategy concurrent debug true
SMI Cluster Manager# clusters test1 actions sync logs
SMI Cluster Manager# monitor sync-logs test1
SMI Cluster Manager# clusters test1 actions sync status
```

Once cluster sync is successful, verify the cluster sync log.

Sample output:

```
Monday 27 October 2025 10:20:53 +0000 (0:00:01.467) 0:24:13.515 ********

2025-10-27 10:20:53.862 DEBUG cluster_sync.m13-cnaaa: Cluster sync successful
2025-10-27 10:20:53.864 DEBUG cluster_sync.m13-cnaaa: Ansible sync done
2025-10-27 10:20:53.864 INFO cluster_sync.m13-cnaaa: _sync finished. Opening lock
2025-10-27 10:20:53.864 INFO cluster_sync.m13-cnaaa: Aggregated sync time: 00:24:25.265
```

NOTES:

- **clusters** *cluster_name*—Specify the information about the nodes to be deployed. *cluster_name* is the name of the cluster.
- **actions**—Configures the actions performed on the cluster.
- **sync run**—Triggers the cluster synchronization.
- **sync logs**—Displays the current cluster synchronization logs.
- **sync status**—Displays the current status of the cluster synchronization.
- **debug true**—Enters the debug mode.
- **monitor sync logs** – Monitors the cluster synchronization process.



Important You can view the pod details after the upgrade through CEE Ops Center. For more information on pod details, see Viewing the Pod Details section.

Verify the Helm Status

This section describes the procedure involved in verifying the helm status. You need to determine whether the deployed helm chart is listed in the helm list successfully.

To determine the helm status:

1. Run the following on the master node to view the list of deployed helm charts.

   helm list -A

   Sample output:

   ```
   cloud-user@m13-cnaaa-master-3:~$ helm ls -A| grep cpc
   cpc-m13-cnat-cps-infrastructure         cpc-m13         1               2025-10-29
   06:24:22.68282772 +0000 UTC  deployed
   cnat-cps-infrastructure-0.6.10-dev-cpc-2025-03-0051-250811004416-4daacd0
   BUILD_2025.03.1.i54
   cpc-m13-cps-radius-ep                   cpc-m13         1               2025-10-29
   06:24:22.681193924 +0000 UTC deployed
   cps-radius-ep-0.6.43-dev-cpc-2025-03-0317-250812071800-2db928a
   BUILD_2025.03.1.i54
   ```

```
cpc-m13-network-query                   cpc-m13         1               2025-10-29
06:24:22.681469304 +0000 UTC deployed
network-query-0.5.4-dev-cpc-2025-03-0095-250716052025-833caab
BUILD_2025.03.1.i54
cpc-m13-ops-center                      cpc-m13         10              2025-10-29
06:18:10.184343814 +0000 UTC deployed
cpc-ops-center-0.6.32-dev-cpc-2025-03-0558-250716054741-99e44f1
BUILD_2025.03.1.i54
cpc-m13-cpc-config                      cpc-m13         1               2025-10-29
06:24:22.68490608 +0000 UTC  deployed
cpc-config-0.6.3-dev-cpc-2025-03-0030-250716051310-045be1e
BUILD_2025.03.1.i54
cpc-m13-cpc-dashboard                   cpc-m13         1               2025-10-29
06:24:22.686188032 +0000 UTC deployed
cpc-dashboard-0.2.17-dev-cpc-2025-03-0196-250716051353-871e51f
BUILD_2025.03.1.i54
cpc-m13-cpc-engine-app-production-rjio  cpc-m13         1               2025-10-29
06:24:22.681441532 +0000 UTC deployed
cpc-engine-app-0.9.2-dev-cpc-2025-03-0788-250811003555-7830316
BUILD_2025.03.1.i54
cpc-m13-cpc-oam-app                     cpc-m13         1               2025-10-29
06:24:22.681257099 +0000 UTC deployed
cpc-oam-app-0.6.2-dev-cpc-2025-03-0022-250716051638-582c2b0
BUILD_2025.03.1.i54
cpc-m13-cpc-services                    cpc-m13         1               2025-10-29
06:24:22.681354868 +0000 UTC deployed
cpc-services-0.6.17-dev-cpc-2025-03-0089-250716051336-a203202
BUILD_2025.03.1.i54
cloud-user@m13-cnaaa-master-3:~$
```

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

   Sample output:

```
[m13-cnaaa/m13] cpc# show running-config helm
Wed Oct  29 06:37:41.475 UTC+00:00
helm default-repository base-repos
helm repository base-repos
 url https://charts.2002-10-192-1--21.sslip.io/cpc.2025.03.1.i54
exit
[m13-cnaaa/m13] cpc#


[m13-cnaaa/m13] cpc# show helm charts version
Wed Oct  29 06:36:48.650 UTC+00:00
CHART                   INSTANCE                                    VERSION
-------------------------------------------------------------------------------------
cpc-ops-center          cpc-m13-ops-center                          BUILD_2025.03.1.i54
cnat-cps-infrastructure  cpc-m13-cnat-cps-infrastructure            BUILD_2025.03.1.i54
cps-radius-ep           cpc-m13-cps-radius-ep                       BUILD_2025.03.1.i54
etcd-cluster            cpc-m13-etcd-cluster                        BUILD_2025.03.1.i54
network-query           cpc-m13-network-query                       BUILD_2025.03.1.i54
ngn-datastore           cpc-m13-ngn-datastore                       BUILD_2025.03.1.i54
cpc-config              cpc-m13-cpc-config                          BUILD_2025.03.1.i54
cpc-dashboard           cpc-m13-cpc-dashboard                       BUILD_2025.03.1.i54
cpc-engine-app          cpc-m13-cpc-engine-app-production-rjio      BUILD_2025.03.1.i54
cpc-oam-app             cpc-m13-cpc-oam-app                         BUILD_2025.03.1.i54
cpc-services            cpc-m13-cpc-services                        BUILD_2025.03.1.i54

[m13-cnaaa/m13] cpc# show helm charts status
Wed Oct  29 06:36:54.252 UTC+00:00
CHART                   INSTANCE                                    STATUS
```

```
--------------------------------------------------------------------------------
cpc-ops-center           cpc-m13-ops-center                       deployed
cnat-cps-infrastructure  cpc-m13-cnat-cps-infrastructure          deployed
cps-radius-ep            cpc-m13-cps-radius-ep                    deployed
etcd-cluster             cpc-m13-etcd-cluster                     deployed
network-query            cpc-m13-network-query                    deployed
ngn-datastore            cpc-m13-ngn-datastore                    deployed
cpc-config               cpc-m13-cpc-config                       deployed
cpc-dashboard            cpc-m13-cpc-dashboard                    deployed
cpc-engine-app           cpc-m13-cpc-engine-app-production-rjio   deployed
cpc-oam-app              cpc-m13-cpc-oam-app                      deployed
cpc-services             cpc-m13-cpc-services                     deployed
[m13-cnaaa/m13] cpc#
```

Verify the Pods

Follow these steps to determine the pod and container status after upgrading cnAAA. Ensure that all pods and containers are up and running.

- Use this command to view the cnAAA pod logs:

  kubectl describe pod *pod_name* **-n** *namespace*

- Use this command to list all pods and filter for those not in a `Running` status.

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

Note 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.

Rollback the upgrade

The upgrade can be rolled back if issues are encountered during the upgrade process. This section describes the procedure for rolling back the upgrade..

Reload cnAAA Ops Center configuration

To reload the cnAAA Ops Center configuration from the backup file, follow these steps:

1. Log in to the SMI console as an **ubuntu** user.
2. Untar the backup file created on SMI and move it into a directory.

   Example:

   ```
   cd ~/backups && tar -zxf popcf-cfg-backup_110219-053530.tar.gz
   ```

3. Move the backup configuration file into the newly created **backups** directory.

   Example:

   ```
   cd popcf-cfg-backup_110219-053530
   ```

4. Convert the exported cnAAA Ops Center configuration into a clean file, which is ready for import.

   Example:

```
cat pcfops*.cfg | perl -pe 's/vendor.*\[(.*)\]/vendor $1/g' | perl -pe
's/(^\s+ips).*\[(.*)\]/$1$2/g' | perl -pe 's/(\w)\s+(\w)/$1 $2/g' | perl -pe 's/^\s+//g'
 | grep -v "system mode run" > cpcops.txt
```

Update cnAAA Ops Center configuration

This section describes the procedure involved in updating the cnAAA Ops Center configuration after restoring it. To update the cnAAA Ops Center configuration:

1. Log in to the master node as an **ubuntu** user.
2. Run the following command to log in to the cnAAA Ops Center CLI.

   Example:

   ```
   ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces | grep -oP
   'cnAAA-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }')
   ```

3. Paste the contents of the exported cnAAA configuration file (the **cnAAAops.txt** file mentioned in the cnAAA Ops Center.

   Example:

   ```
   product cnAAA# config
   Entering configuration mode terminal
   product cnAAA(config)# <PASTE CONTENTS OF cnAAAops.txt AND RETURN TO 'config' mode. Don't
    Paste Default Configuration>
   product cnAAA(config)#
   ```



Important Fix any sections in the configuration file that did not import properly.

4. Ensure that the helm URLs are inline with the updated cnAAA image.

   Example:

   ```
   product cnAAA(config)# helm repository base-repos
   product cnAAA(config-repository-base-repos)# url <url>
   product cnAAA(config-repository-base-repos)# exit
   product cnAAA(config)# k8s registry <registry_url>
   product cnAAA(config)# commit
   Commit complete.
   product cnAAA(config)#
   ```

Restore the configuration from backup

This section describes the procedure involved in restoring all the Policy Builder and CRD configuration files from the backup.

Restore policy builder configuration

1. Log in to the master node as an **ubuntu** user.
2. Retrieve the Cisco Policy Suite Central URL.

Example:

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

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

3. Navigate to the Cisco Policy Suite Central URL and log in with user credentials.
4. Click **Import/Export** and select the **Import** tab.
5. Select the exported policy backed up in the Back Up SVN, Policy, and CRD Data section.
6. In **Import URL**, specify the following URL:

   `http://svn/repos/configuration`
7. Enter a brief description in **Commit Message** and click **Import**to commit the changes in Policy Suit Central.
8. Log in to the master node again as an **ubuntu** user.
9. Run the following command to retrieve the Cisco Policy Builder URL.

   Example:

```
kubectl get ing -n $(kubectl get namespaces | grep -oP 'cnAAA-(\d+|\w+)' | cut -d\
-f1) | grep policy-builder | awk '{ print "https://"$2"/pb" }'
https://pb.cnAAA-02-cnAAA-engine-app-blv02.<ipv4address>.nip.io/pb
ubuntu@pocnAAA-mas01:~/backups_09182019_T2141$
```

10. Navigate to the Cisco Policy Builder URL and click **Build Policies using version controlled data** .
11. Choose **Repository** from the drop-down list and click **OK**.
12. Click **Publish to Runtime Environment** and enter a brief description in **Commit Message**.
13. Click **OK.**

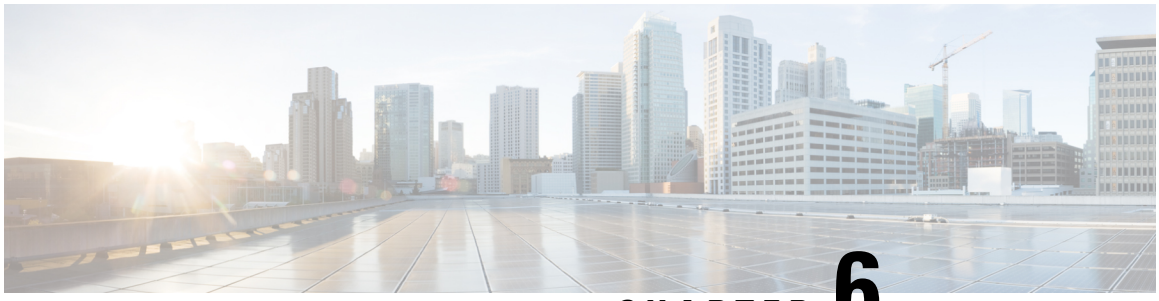
Restoring CRD Data

1. In CPS Central home page, click **Custom Reference Data**.
2. Check the **Export CRD to Golden Repository** check-box.
3. Specify the SVN host name in **Please enter valid server Hostname or IP** text-box.

Note For cnAAA the SVN host name value is *svn*.

4. Click +.
5. Click **Export**.

Note You receive a success message when the data is exported successfully.



CHAPTER 6

Subscriber Profile Repository

- Support for Mongo and CDL, on page 85
- MongoDB authentication, on page 88
- Subscriber migration to multiple SPR database replica sets, on page 95
- Replica set for admin DB similar to SPR for sharing across clusters, on page 97
- Support for multiple SPR replica sets, on page 100
- Multiple arbiter qualification for SCDB, on page 108
- Consistent replica set routing in apiRouter under high TPS, on page 110
- Configurable MongoDB storage size, on page 114
- Auto scheduler backup for MongoDB, on page 116
- Configure the MongoDB replica set, on page 116
- Configure global backup settings, on page 117
- Perform on-demand backups and monitoring, on page 117
- Restore the SCDB SPR backup, on page 118
- Troubleshoot the local backup safety net, on page 119
- Subscriber authentication and Called-Station-ID uniqueness, on page 119

Support for Mongo and CDL

Feature History

Feature Name	Release Information	Description
Subscriber migration to multiple SPR DB replica sets	2025.02.0	This feature migrates data from CPS 7.5 to CPC. distributing records across multiple SPR databases. It extracts data from CPS 7.5, restores it to SPR1 in CPC, and assigns hash values for subscriber distribution across SPR1, SPR2, and SPR3.

Feature Name	Release Information	Description
Mongo 7.0 Support for Subscriber Profile Repository (SPR)	2025.01.0	cnAAA provides support for storing Subscriber Profile Repository (SPR) information in Mongo 7.0. The Subscriber information can be created, updated, or deleted through SOAP unified API calls. Mongo 7.0 is deployed as a replica set with replica set members deployed on local and remote sites. Mongo DB ensures that the data is replicated across sites, ensuring geo redundancy and availability in case of failure on any of the sites.

Overview

cnAAA provides support for storing Subscriber Profile Repository (SPR) information in Mongo 7.0. The subscriber information can be created, updated, or deleted through SOAP unified API calls.

Mongo 7.0 is deployed as a replica set with replica set members deployed on local and remote sites. Mongo DB ensures that the data is replicated across sites, ensuring geo redundancy and availability in case of failure on any of the sites.

Configure of the SPR MongoDB replica set

Install the SPR MongoDB replica set within the cnAAA Kubernetes cluster namespace. Configure the replica member hosts to have network reachability to the cnAAA Engine Pods, using node IPs of master nodes as replica member host IP addresses.

SPR DB replica set configurations

- MongoDB replica set members utilize the host network to facilitate network connectivity across different sites.
- Ensure that host network reachability is maintained across mated pair sites to prevent issues with database initialization and replication.

Configure Policy Builder and remote database with USuM configuration

To set up the Policy Builder and configure the remote Subscriber Profile Repository (SPR) database using the USuM configuration tab, follow these steps:

Configure Policy Builder

Set up the Policy Builder with primary and secondary replica members, including their port numbers, under **Plugin Configurations** > **USuM Configuration**.

Configure Remote Database with USuM Configuration

Procedure

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

Step 2 **Select USuM Configuration**: Choose the **USuM Configuration** tab.

Step 3 **Enter Database Values**: Input the mandatory database values required for configuration.

Configure the MongoDB tuning

To optimize MongoDB performance for cnAAA, configure the these database tuning parameters with required values.

```
db
 global-settings
 db-tunings
 oplog-size-mb 3072
 wired-tiger-cache-size-gb 3
 slowms 500
 concurrent-transactions-read 2
 concurrent-transactions-write 3
```

Notes:

oplog-size-mb - 3072 indicates 3GB space allocated for oplog.

wired-tiger-cache-size-gb - Defines the maximum size of the internal cache that WiredTiger uses for all data.

slowms - The slow operation time threshold, in milliseconds.

concurrent-transactions-read - Specify the maximum number of concurrent read transactions (read tickets) allowed into the storage engine.

concurrent-transactions-write - Specify the maximum number of concurrent write transactions (write tickets) allowed into the storage engine.

Note Mongo authentication is currently not supported in this release.

Configure Mongo Replica Sets

Prerequisite: Before configuring the replica set, create the labels for the nodes.

To Configure Mongo replica set configurations using cnAAA, follow these steps:

- **Provision Mongo Replica Sets:**Set up Mongo replica sets by configuring their settings and assigning database labels on Kubernetes nodes to ensure replica set members spawn on the appropriate master nodes.
- **Sample Configuration:**This configuration sets up a database replica with an arbiter and two members, specifying their roles, priorities, and network settings.

```
db scdb replica-name sdb-subscriber1
 port      65001
 interface vlan2400
 resource cpu limit 6000
 resource memory limit 112000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs1-arbiter
  host    10.192.1.24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs1-s1-m1
  host     10.192.1.22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs1-s1-m2
  host     10.192.1.23
  arbiter  false
  priority 101
  site     local
 exit
exit
```

For more information on replica sets, see *Mobile Policy Common Commands* in *Ultra Cloud Core Policy and Charging, Release 2025.01.0- CLI Reference Guide.*

Verification

You can verify the installation of SPR using the following command:

```
show db scdb replica-set-status
```

- **Verify Replica Set Status:** Ensure the replica set initializes correctly, with each member assigned the correct role (primary, secondary, arbiter).

```
cloud-user@alpha-master-1:~$ kubectl get pods -A|grep db-scdb
<namespace>         db-scdb-sdb-subscriber1-0                                        1/1
    Running   0              7d5h
<namespace>         db-scdb-sdb-subscriber1-1                                        1/1
    Running   0              7d5h
<namespace>         db-scdb-sdb-subscriber1-2                                        1/1
    Running   0              7d5h
cloud-user@alpha-master-1:~$
```

MongoDB authentication

Introduction to MongoDB authentication

The Converged Policy and Charging (CPC) uses MongoDB as its administrative database. MongoDB supports critical functionalities, such as Custom Reference Data (CRD), Subscriber Profile Repository (SPR), and Policy tracing. To ensure robust security, CPC uses the MongoDB authentication method. This approach provides end-to-end protection for these databases.

The key benefits are:

- Security: Enabling authentication protects sensitive database information.
- Configuration: Simple configuration options improve ease of use for MongoDB.
- Encryption: Passwords are encrypted for secure storage and usage.

MongoDB authentication process

Perform MongoDB authentication process in three stages:

- Authentication setup
- Password Encryption
- Database connection

Authentication setup

You can perform these authentication to:

- Enable MongoDB authentication using the Ops-Center configuration commands
- Configure the MongoDB username and passwords

Password encryption

To perform password encryption:

- Use the CLI utility to encrypt the password.
- Add the encrypted password as a JVM argument for the Engine, PB, and CRD processes.

Database connection

When you connect to the database:

- Ensure reachability between MongoDB replica members and CPC Engine pods.
- Access MongoDB using the mongo command with the actual username and password.

Enable database authentication

Before you begin

Enable MongoDB authentication to secure subscriber-specific, Session Control Database (SCDB), and administrative configuration data.

- Enable MongoDB authentication only when the system is in a shutdown state.
- If authentication is enabled in CPC but not enabled in CPS, the system may fail to integrate or migrate.
- Configure the cdl-layer and protocol-layer labels in Ops-Center using the key properties.

- label cdl-layer key smi.cisco.com/node-type-3 value session.
- label protocol-layer key smi.cisco.com/node-type-2 value protocol.

- Use the same MongoDB username and password across all sites.

Complete these steps to enable and verify database authentication:

Procedure

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

Step 2 Configure the global database credentials.

```
db global-settings
   db-user-name <MongoDB_username>
   password <PlainTextPassword>
```

Example:

```
db global-settings db-user-name admin password abcxy@123
```

Step 3 `commit` the changes to enable the MongoDB authentication.

Step 4 Log in to the Ops-Center configuration.

The engine name is displayed according to the deployment configurations.

Example:

engine *name* **properties** *name* **value** *value*

Step 5 Enter the **ops-center** command to encrypt or decrypt the plain text password:

a) Enter the **db scdb execute password encrypt-password plain-text***PlainTextPassword* command.

Example:

```
[unknown] pcf# db scdb execute password encrypt-password plain-text abcxy@123
Mon Jun  16 08:14:14.138 UTC+00:00
output :
3300901EA069E81CE29D4F77DE3C85FA
```

Example:

Note Use the password specified when enabling the MongoDB authentication parameter.

b) Enter the **db scdb execute password decrypt-password encrypted-text** *EncryptedPassword* command to decrypt the password.

Example:

```
[unknown] pcf# db scdb execute password decrypt-password encrypted-text
3300901EA069E81CE29D4F77DE3C85FA
Fri Jun 16 08:14:14.138 UTC+00:00
output :
abcxy@123
```

Step 6 Enter the properties required to set the database username and encrypted password for the engine using these commands:

```
engine engine_name properties dbUsername value username
 engine engine_name properties dbPassword value encryptedpassword
 exit
```

Example:

```
engine pcf-green properties dbUsername value admin
engine pcf-green properties dbPassword value 3300901EA069E81CE29D4F77DE3C85FA
```

Step 7 `commit` the changes.

What to do next

After you enable MongoDB authentication, complete these tasks:

- Verify the connectivity to MongoDB replica members from CPC engine pods.
- Access the MongoDB replica set by connecting to the primary host IP and port with the admin username and specified password.

  ```
  mongo primary_host_ip:mongo_port -u admin -p password
  ```

Sample output to verify the subscriber count without MongoDB authentication credentials:

```
cloud-user@alpha-master-1:~$ kubectl
exec -it <pod name> -n <namespace>
-c mongo -- env HOME=/tmp mongo --host <host ip address>
--port <port number> --quiet --eval "db = db.getSiblingDB('spr');
db.subscriber.countDocuments()"
MongoServerError: Command aggregate requires authentication
command terminated with exit code 1
```

Sample output to verify the subscriber count with MongoDB authentication credentials:

```
cloud-user@alpha-master-1:~$ kubectl exec -it
<pod name> -n <namespace> -c mongo -- env HOME=/tmp mongo --host <host ip address>
--port <port number> -u <user name> -p <password>
--quiet --eval "db = db.getSiblingDB('spr'); db.subscriber.countDocuments()"
1537280
```

Note This feature cannot be enabled on the GR site. Enabling it requires a fresh installation with Mongo authentication.

MongoDB backup user privileges

This is a security feature that allows to create user accounts with limited permissions. These users can only perform database backup operations on the MongoDB database. They cannot perform any other administrative or operational tasks. This helps prevent unauthorized changes and accidental data loss by backup-only users.

Prerequisites for Configuring Backup-Only User Accounts

Before creating a backup-only user account, ensure that:

- Service Control Database (SCDB) replicas are configured.
- Persistent Volume is configured.
- The Kubernetes setting "use-volume-claims" is enabled and set to true in the Ops-Center.
- Remote server details for backup storage are configured.

Configure Backup-Only user accounts

Procedure

Follow these steps to configure a backup-only user account:

Step 1 Set up the remote server to store MongoDB backups and specify the host, port, username, password, and remote path for backup files.

Sample configuration:

```
db global-settings backup-settings scp-server host <remote server ip>
db global-settings backup-settings scp-server port 22
db global-settings backup-settings scp-server user-name <username>
db global-settings backup-settings scp-server password <password>
db global-settings backup-settings scp-server remote-backup-path <path for storing backup file>
```

Step 2 Create a backup user and assign to backup group:

```
smiuser add-user username <username> password <password>
smiuser add-group groupname <group_name>
smiuser assign-user-group username <username> groupname <group_name>
```

Example:

```
smiuser add-user username backup_user password Cisco@123
smiuser assign-user-group username backup_user groupname backup
smiuser list-users
```

Note Use the smiuser list-users command to verify user and group assignment.

Step 3 Configure Network Access Control Model (NACM) Rules

a) Apply deny rules to the backup group so that the user can only perform the backup operation.

```
Deny rules:
nacm rule-list crdread group <group_name> cmdrule denyaaa command aaa access-operations exec action
 deny
 nacm rule-list crdread group <group_name> cmdrule denycd command cd access-operations exec action
 deny
 nacm rule-list crdread group <group_name> cmdrule denyclear command clear access-operations exec
 action deny
 nacm rule-list crdread group <group_name> cmdrule denycommit command commit access-operations
exec action deny
 nacm rule-list crdread group <group_name> cmdrule denycompare command compare access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfig command config access-operations
exec action deny
 nacm rule-list crdread group <group_name> cmdrule denydeployment command deployment
```

```
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denydescribe command describe access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denydiameter-peer command diameter-peer
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyhelp command help access-operations exec
action deny
 nacm rule-list crdread group  <group_name> cmdrule denyhistory command history access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyid command id access-operations exec action
 deny
 nacm rule-list crdread group <group_name> cmdrule denyidle-timeout command idle-timeout
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyignore-leading-space command
ignore-leading-space access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyjob command job access-operations exec
action deny
 nacm rule-list crdread group <group_name> cmdrule denyleaf-prompting command leaf-prompting
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denylicense command license access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyno command no access-operations exec action
 deny
 nacm rule-list crdread group <group_name> cmdrule denypaginate command paginate access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denypcf-system command pcf-system
access-operations exec action deny
 nacm rule-list crdread group  <group_name> cmdrule denyscreen-length command screen-length
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyscreen-width command screen-width
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denysend command send access-operations exec
action deny
 nacm rule-list crdread group <group_name> cmdrule denyshow command show access-operations exec
action deny
 nacm rule-list crdread group <group_name> cmdrule denyshow-defaults command show-defaults
access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denysmildap command smildap access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denysmiuser command smiuser access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denysystem command system access-operations
exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyterminal command terminal access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denytimestamp command timestamp access-operations
 exec action deny
 nacm rule-list crdread group <group_name> cmdrule denywho command who access-operations exec
action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfigaddmem command "db scdb execute
add-member" access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfigdisablemon command "db scdb execute
 disable-mongo-flow-control" access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfigenablemon command "db scdb execute
enable-mongo-flow-control" access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfigremovemem command "db scdb execute
remove-member" access-operations exec action deny
 nacm rule-list crdread group <group_name> cmdrule denyconfigrestoredata command "db scdb execute
 restore-data" access-operations exec action deny
```

b) These commands are denied for the backup group:

```
aaa, cd, clear, commit, compare, config, deployment, describe, diameter-peer,
 help, history, id, idle-timeout, ignore-leading-space, job, leaf-prompting, license, no,
```

```
paginate, pcf-system, screen-length, screen-width,
send, show, show-defaults, smildap, smiuser, system, terminal, timestamp, who.
```

c) These database commands are denied for backup user:

```
db scdb execute add-member
db scdb execute disable-mongo-flow-control
db scdb execute enable-mongo-flow-control
db scdb execute remove-member
db scdb execute restore-data
```

Example:

```
[gamma/gamma-cncps] pcf# db scdb execute disable-mongo-flow-control
Wed Apr 16 00:23:36.455 UTC+00:00 Aborted: permission denied
```

Note Permitted database command for the backup user is `db scdb execute backup-data`.

Verify Backup-Only user account

Procedure

Step 1 Use Secure Shell (SSH) to log in with the backup user's credentials.

Example:

```
ssh backup_user@10.103.125.143 -p 2024
```

Step 2 Run *smiuser ?* to view permitted commands.

Step 3 Execute the *db scdb execute backup-data* command in the Ops center to start a backup.

Example:

```
[gamma/gamma-cncps] pcf# db scdb execute backup-data
Wed Apr  16 00:21:01.291 UTC+00:00
output
Backup initiated
```

Step 4 Log in to the remote server and verify the backup file exists in the specified directory.

Example:

```
ls
mongodb_backup-2025-04-15-11-01-07.tar.gz
```

Subscriber migration to multiple SPR database replica sets

Overview

This feature migrates subscriber data from CPS 7.5 to CPC, distributing records across multiple Subscriber Profile Repository (SPR) databases to optimize the data management. Subscriber data is extracted from CPS 7.5 and restored to SPR1 within CPC. A custom tool assigns hash values to these records, and distributes to SPR2 and SPR3. Duplicate records are removed from each SPR database, based on their assigned hash value. This migration improves performance, scalability, and data management in the CPC.

Data Migration from CPS 7.5 to CPC

This process outlines the steps for migrating subscriber data across multiple SPR databases in cnAAA.

Procedure

Step 1 Extract Subscriber Data from CPS 7.5.

Perform a MongoDB dump to back up subscriber data from CPS 7.5.

```
mongodump --host sessionmgr02 --port 27720 --out /root/sprdump
```

Step 2 Restore Data to SPR1 in CPC.

Log in to SPR1 DB Pod and use MongoDB restore to import the subscriber dump into SPR1.

```
mongorestore --host <primary replica member IP> --port 65001 --db spr --verbose SPRDUMP-5M/dump/spr/
```

Step 3 Run the migration tool.

Log in to the engine pod and execute the following commands:

- ```
  mkdir /tmp/1
  cd /tmp/1
  jar -xf /app/plugins/com.google.guava_*.jar
  jar -xf /app/plugins/org.mongodb.mongo-java-driver_*.jar
  jar -xf /app/plugins/com.broadhop.spr.migration.networkId.hashing_2025.2.1.r0bb3cc9f0.jar
  jar -cfm hashing_2025.2.1.r0bb3cc9f0.jar META-INF/MANIFEST.MF *
  ```

- Use the tool.

  ```
  java -jar hashing_2025.2.1.r0bb3cc9f0.jar <primary replica member IP> <sdb-spr1 db port> <maxHash>
  ```

Note Get the latest jar file name under `/app/plugins` path inside the engine pod.

Step 4 Export from SPR1 and import to SPR2 and SPR3.

Execute mongoexport command on SPR1 to collect mongo dump.

```
mongoexport --host <primary replica member IP> --port 65001 --db spr --collection subscriber --out
subscriberMigration_allSub.json
```
```

Use mongoimport to import the exported dump on both SPR2 and SPR3.

```
mongoimport --host <primary replica member IP> --port 65002 --db spr --collection subscriber
subscriberMigration_allSub.json
mongoimport --host <primary replica member IP> --port 65003 --db spr --collection subscriber
subscriberMigration_allSub.json
```



Note Execute the mongo export and mongo import commands under `/data/db` path inside the `br-controller-scdb` pod.

Step 5 Cleanup Duplicate Records.

a. Connect to the spr01 DB and remove the records whose hash is not 0.

```
mongo --host <primary replica member IP> --port 65001

use spr

db.subscriber.deleteMany({avps_key: {$elemMatch: {code_key: "_SprKeyHash",value_key: { $ne: "0"
}}}})
```

b. Connect to the spr02 DB and remove the records whose hash is not 1.

```
mongo --host <primary replica member IP>  --port 65002

use spr

db.subscriber.deleteMany({avps_key: {$elemMatch: {code_key: "_SprKeyHash",value_key: { $ne: "1"
}}}})
```

c. Connect to the spr03 DB and remove the records whose hash is not 2.

```
mongo --host <primary replica member IP>  --port 65003

use spr

db.subscriber.deleteMany({avps_key: {$elemMatch: {code_key: "_SprKeyHash",value_key: { $ne: "2"
}}}})
```

Step 6 Verify subscriber count.

Verify that the total count of SPR1, SPR2, and SPR3 in CPC matches the original SPR1 count from CPS 7.5.

a. Connect to the spr01 DB and get the subscriber count.

```
mongo --host <primary replica member IP>  --port 65001

use spr

db.subscriber.count()
```

b. Connect to the spr02 DB and get the subscriber count.

```
mongo --host <primary replica member IP>  --port 65002

use spr

db.subscriber.count()
```

c. Connect to the spr03 DB and get the subscriber count.

```
mongo --host <primary replica member IP>  --port 65003

use spr
```

```
db.subscriber.count()
```

Replica set for admin DB similar to SPR for sharing across clusters

Feature History

Feature Name	Release Information	Description
Replicaset for admin DB similar to SPR for sharing across clusters	2025.03.0	This feature adds a dedicated admin DB replica set in the CPC namespace to manage Customer Reference Data (CRD) operations, isolating CRD traffic from other system processes and enhancing data consistency, access efficiency, and redundancy.

Overview

This feature adds a dedicated admin DB replica set in the CPC namespace. The admin DB replica set manages CRD operations, such as importing and exporting tables. This isolates CRD traffic from other system processes and prevents interference with existing configurations. Using a separate replica set with a unique port improves data consistency, access efficiency, and provides redundancy for continuous CRD availability.

Label nodes for the admin DB replica set configuration

Procedure

Step 1 Log in to a master node and enter this command to check the existing node labels:

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

Step 2 Login to the cluster manager and access Ops Center CLI.

Step 3 Enter configuration mode:

```
config
```

Step 4 Enter the cluster configuration:

```
cluster <cluster-name>
```

Step 5 Add the node label for each master node:

```
nodes master-1
k8s node-labels smi.cisco.com/node-type-5 admin-db
exit
nodes master-2
k8s node-labels smi.cisco.com/node-type-5 admin-db
exit
nodes master-3
k8s node-labels smi.cisco.com/node-type-5 admin-db
exit
exit
```



Note Replace the <*cluster-name*> placeholder with the actual cluster name for each master node to be labeled.

Step 6 Sync the cluster configuration:

```
clusters <cluster-name> actions sync run upgrade-strategy concurrent debug true
```

Step 7 Monitor the synchronization logs:

```
monitor sync-logs <cluster-name>
```

Configure admin DB Replica Sets

To Configure admin DB replica set, follow these steps:

```
db scdb replica-name admin-db
 port      65005
 interface vlan2400
 resource cpu limit 3000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type-5
 replica-set-label value admin-db
 member-configuration member sdb-rs4-s1-arbiter1
  host    192.0.2.44
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs4-s1-m0
  host    192.0.2.32
  arbiter false
  priority 104
  site    local
 exit
 member-configuration member sdb-rs4-s1-m1
  host     192.0.2.33
  arbiter  false
  priority 103
  site     local
 exit
exit
```

Configure CRD API properties in Ops Center

Configure the CRD API properties in Ops-Center to enable the engine and other components to connect with the admin DB replica set.

Procedure

Step 1 Login to Ops-Center and enter the `config` command.

Step 2 Enter the CRD API properties for admin DB:

```
engine <engine-name> crdapi admin-db primary <Primary-member-IP> secondary <Secondary-member-IP> port
 <DB-PORT>
```

Step 3 Repeat the configuration command on all remote sites.

Step 4 Verify the configuration by entering the `show running-config engine` command.

Note Confirm that the `crdapi admin-db` entries for primary, secondary, and port are correct.

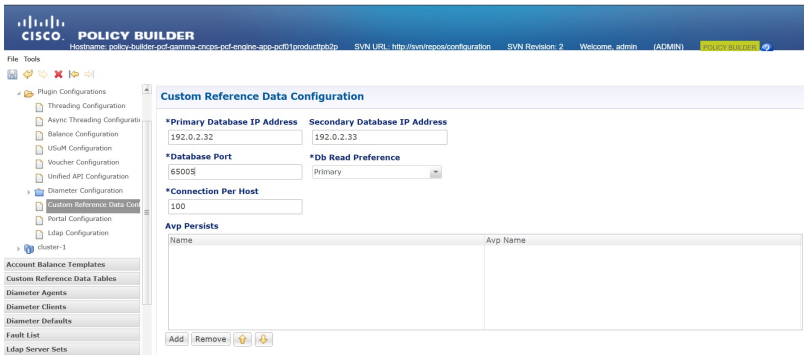
Step 5 Restart the engine and CRD API pods in all affected clusters.

Example

```
[unknown] pcf# show running-config 'engine.<user_configurable_term>' <user
crdapi admin-db primary 192.0.2.32 secondary 192.0.2.33 port 65005
```

Policy Builder configuration

Follow these steps to update the admin DB connection settings in Policy Builder.



Procedure

Step 1 Navigate to **Plugin Configuration** > **Custom Reference Data Configuration**.

Step 2 Enter the Primary IP address of the admin DB replica set.

Step 3 Enter the Secondary IP address of the admin DB replica set.

Step 4 Enter the port number for the admin DB connection.

Step 5 Publish the Policy Builder and check the pod status.

Note These fields will now use IP addresses and port numbers instead of hostnames to use the latest Policy Builder changes.

Note After completing the configuration, restart all engine and CRD API pods in the Kubernetes cluster to reflect the latest Policy Builder changes.

Support for multiple SPR replica sets

Feature History

Feature Name	Release Information	Description
Support for Multiple SPR Replica Sets	2025.02.0	This feature distributes subscriber details evenly across SPR Mongo replica sets. An enhanced API Router uses a network ID-based hashing mechanism for Radius subscribers to manage record distribution.

Overview

Support for multiple SPR replica sets in cnAAA distributes subscriber records across Mongo replica sets to improve performance in handling subscriber data. The API Router has been enhanced with a network ID-based hashing mechanism to manage Radius subscribers.

Configure Subscriber Database Replica Sets

Configure the Scaling SPR Mongo replica sets to distribute subscriber details.

Procedure

Step 1 **Replica Set Configuration**

a) Set Ports and Interface

```
Configure the port for sdb-subscriber1 to 65001
Configure the port for sdb-subscriber2 to 65002
Configure the port for sdb-subscriber3 to 65003
Set the interface to vlan2400
```

b) Resource Limits

```
Set CPU limit to 6000
Set memory limit to 112000
```

c) Replica Set Label

```
Set key to smi.cisco.com/node-type.
Set value to oam.
```

Step 2 **Member Configuration for sdb-subscriber1**

a) Arbiter Configuration

```
Add member sdb-rs1-arbiter with the following settings:
Host: 10.192.1.24
Arbiter: true
Site: local
```

b) Data Member Configuration

```
Add member sdb-rs1-s1-m1 with the following settings:
Host: <Primary database host ip>
Arbiter: false
Priority: 102
Site: local
Add member sdb-rs1-s1-m2 with the following settings:
Host: <Secondary database host ip>
Arbiter: false
Priority: 101
Site: local
```

Step 3 **Member Configuration for sdb-subscriber2**

a) Arbiter Configuration

```
Add member sdb-rs2-arbiter with the following settings:
Host: 10.192.1.24
Arbiter: true
Site: local
```

b) Data Member Configuration

```
Add member sdb-rs2-s2-m1 with the following settings:
Host: <Primary database host ip>
Arbiter: false
Priority: 102
Site: local
Add member sdb-rs2-s2-m2 with the following settings:
Host: <Secondary database host ip>
Arbiter: false
Priority: 101
Site: local
```

Step 4 **Member Configuration for sdb-subscriber3**

a) Arbiter Configuration

```
Add member sdb-rs1-arbiter with the following settings:
Host: 10.192.1.24
Arbiter: true
Site: local
```

b) Data Member Configuration

```
Add member sdb-rs3-s3-m1 with the following settings:
Host: <Primary database host ip>
Arbiter: false
Priority: 102
Site: local
Add member sdb-rs3-s3-m2 with the following settings:
Host: <Secondary database host ip>
Arbiter: false
Priority: 101
Site: local
```

MongoDB replica set auto-recovery

This feature resolves situations where a replica set member is stuck in the RECOVERING state which prevents the member from rejoining the replica set and impact database availability and consistency.

These stages describe how the auto-recovery mechanism detects and recovers a MongoDB replica set member in the RECOVERING state:

1. The MongoDB health check script runs every ten seconds as part of the container's liveness and five seconds as part of readiness probes.
2. The script monitors the status of replica set members.
3. If a non-primary member is in the 'RECOVERING' state, the script increments a counter variable.
4. If the counter reaches a predefined threshold (for example, five consecutive checks), the script initiates the recovery process.
5. The recovery process deletes the contents of the /data/db directory on the affected member and restarts the pod.
6. After the restart, the pod re-synchronizes its data from the primary node and rejoins the replica set.

Key components of the auto-recovery mechanism:

- Supports both IPv4 and IPv6 environments.
- Works with MongoDB deployments with or without authentication. If authentication is enabled, the script uses the provided credentials.

Configure auto-recovery timeout for MongoDB replica set

By default the recovery steps will be executed after 120 seconds. You can increase the time accordingly by following these steps:

Procedure

Step 1 Enter the global configuration mode.

Step 2 Run this command to set the timeout

```
db global-settings timers auto-recovery-timeout-secs <seconds>
```

This command sets the maximum time in seconds to wait before initiating recovery for a MongoDB member stuck in the 'RECOVERING' state.

Sample Configuration:

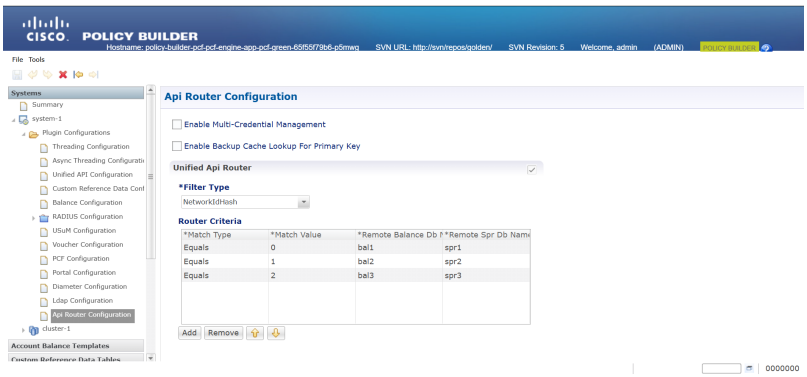
```
[unknown] pcf(config)# db global-settings timers auto-recovery-timeout-secs 200
Thu Jul  17 06:51:57.219 UTC+00:00
[unknown] pcf(config)#
```

Configure the Network ID in API Router

This section describes how to configure the Network ID in the API Router using the network ID hash.

Procedure

Step 1 Set Up the Router Filter for Network ID Hashing



a) Log in to Policy Builder and navigate to Api Router Configuration.
b) Choose Filter Type > NetworkIdHash.

Note Select only NetworkIdHash for the filter type.

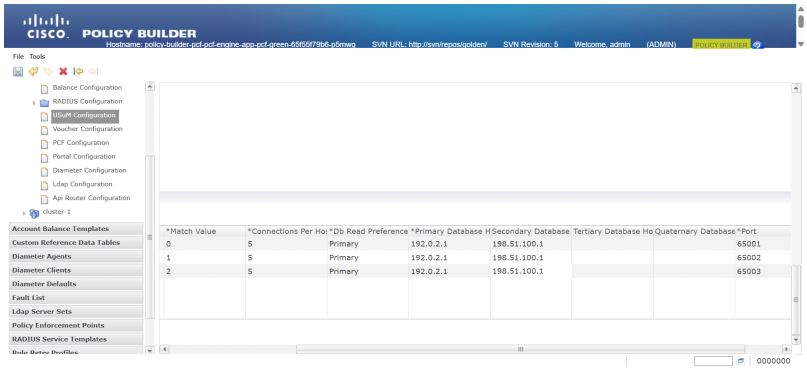
c) Set Router Criteria with the these parameters.

- Match Type
- Match Value
- Remote Balance DB
- Remote SPR Db Name

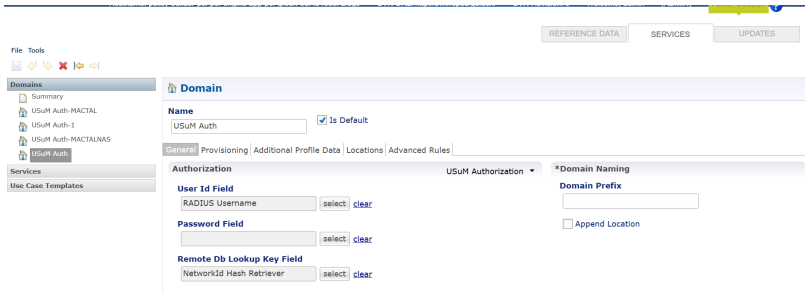
Note Select **Equals** for Router Criteria Match Type.

d) Click **Add**.

Step 2 Add a New Remote Database Configuration



a) Navigate to Plugin Configuration > UsUM Configuration.
b) Select Set Remote Database Configuration.
c) Enter these parameters.



- Name
- Match Type
- Match Value
- Connections Per Host
- DB Read Preference
- Primary Database
- Secondary Database

d) Click **Add**.

Note After enabling the scaling SPR feature, use /apirouter for the SOAP API instead of the current /ua/soap.

Note Select NetworkId Hash Retriever for Remote Db Lookup Key Field.

Configure Ops Center for multiple SPR replica sets

To set up multiple SPR replica sets in cnAAA Ops Center, follow these steps:

Procedure

Step 1 Login to Ops Center.

Step 2 Configure cnAAA Engine Properties

- Set Balance Key Hash AVP Name

  Configure `balanceKeyHashAvpName` with `SprKeyHash` to specify the AVP name used for balance key hashing.

- Set SOAP URL

  Configure `cc.ua.soap.url` with `http://127.0.0.1<host>:8080<port>/apirouter` to define the endpoint for SOAP API requests. Replace `<host>` and `<port>` with the appropriate values for your setup.

- Define Maximum Hash Value

  Set `maxHash` to `2` to specify the maximum hash value allowed.

- Enable Site Query for Subscriber Search

  Configure `queryEachSiteForSearchSubscribers` to `true` to enable site queries when searching for subscribers.

- Configure Balance Return Option

  Set `returnBalance` to `false` to disable balance information return during operations.

- Enable SPR Hash Support

  Set `sprHashSupportEnabled` to `true` to activate subscriber profile repository hash support

- Set `properties replaceFullNameInSearchSubscribers value true` and enter `exit` to enable this configuration to search subscribers through control center based on network ID (optional).

Step 3 Enter `exit` to save changes and exit configuration mode.

Sample Configuration:

```
engine production-rjio
 properties balanceKeyHashAvpName
  value SprKeyHash
 exit
 properties cc.ua.soap.url
  value http://127.0.0.1:8080/apirouter
 exit
 properties maxHash
  value 3
```

```
 exit
 properties queryEachSiteForSearchSubscribers
  value true
 exit
properties returnBalance
  value false
 exit
 properties sprHashSupportEnabled
  value true
 exit
properties replaceFullNameInSearchSubscribers
  value true
 exit
exit

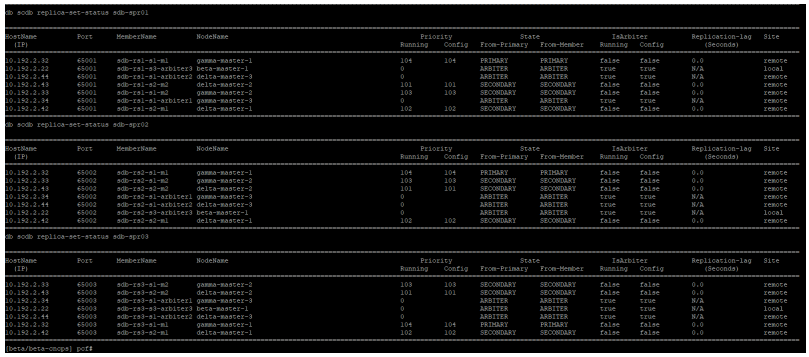
DB-SPR Configuration: 3 Replica Set

db scdb replica-name spr1
 port      65001
 interface vlan2400
 resource cpu limit 6000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs1-arbiter
  host    192.0.2.1
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs1-s1-m1
  host     192.0.2.1
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs1-s1-m2
  host     198.51.100.1
  arbiter  false
  priority 101
  site     local
 exit
exit
db scdb replica-name spr2
 port      65002
 interface vlan2400
 resource cpu limit 6000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs2-arbiter
  host    10.192.1.24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs2-s1-m1
  host     192.0.2.1
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs2-s1-m2
  host     198.51.100.1
  arbiter  false
  priority 101
```

```
  site     local
 exit
exit
db scdb replica-name spr3
 port      65003
 interface vlan2400
 resource cpu limit 6000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs3-arbiter
  host    10.192.1.24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs3-s1-m1
  host     192.0.2.1
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs3-s1-m2
  host     198.51.100.1
  arbiter  false
  priority 101
  site     local
 exit
exit
```

Note The `db scdb replica-name` command reflects the number of SPR replicas (e.g., spr1, spr2, spr3). Specify the desired number of replicas as required.

Step 4 Validate the replica set status on cnAAA Ops Center using the `show db scdb replica-set-status` command.



Multiple arbiter qualification for SCDB

Feature History

Feature Name	Release Information	Description
Multiple Arbiter qualification for SCDB	2025.02.0	This feature enhances the high availability and strength of the Subscriber Configuration Database (SCDB) by configuring multiple arbiters within each replica set. It improves system reliability, ensuring uninterrupted operations in Geo-Redundant deployments.

Overview

This feature enhances the reliability of the SCDB by enabling the setup of multiple arbiters within each replica set. It improves stability, particularly in geographically distributed deployments. With multiple arbiters, the replica set can continue to function and elect a primary even if some arbiters are unavailable. As a result, the system experiences less downtime and provides consistent service.

Ops-Center Configuration for multiple arbiters in a Replica Set

To configure multiple arbiters within an SCDB replica set using the CPC Ops-Center, follow these steps:

Before you begin

Procedure

Step 1 Log in to the CPC Ops-Center.

Step 2 Specify the `replica-name` for the SCDB replica set. (example: `sdb-spr01`).

```
db scdb replica-name sdb-spr01
```

Step 3 Configure the replica set port using an available port from the range 65001 to 65010 and define the network interface.

```
port      <65007>
interface <vlan2400>
```

Step 4 Configure the CPU and memory resource limits for the replica set.

```
 resource cpu limit 3000
 resource memory limit 20000
```

Step 5 Set the replica set labels (key-value pairs).

```
Define db-spr labels for scdb and update
replica-set-label key smi.cisco.com/node-type-5
replica-set-label value db-spr
```



Note Database key-value pair labels can be configured as per the deployments to determine which K8 nodes are selected for Database Pods.



Note Create database (DB) key-value pair labels on the nodes during cluster deployment.

Step 6 Define each member of the replica set and set the host IP address

```
member-configuration member sdb-rs1-s1-arbiter1
 host    <10.192.2.34>
```

Step 7 Set Arbiter to true.

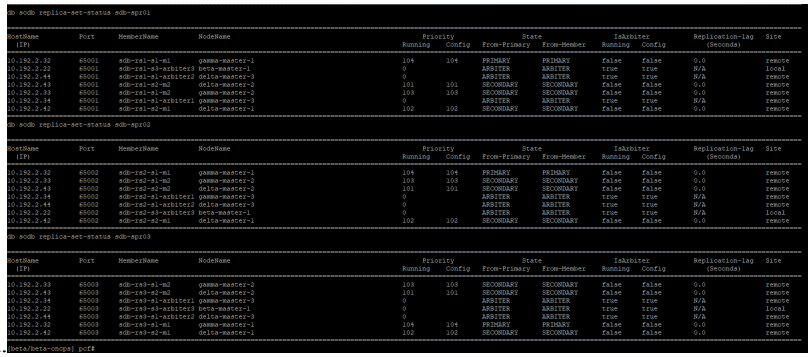
```
 arbiter true
```

```
db scdb replica-name sdb-spr03
 port      65009
 interface vlan2400
 resource cpu limit 3000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value db-spr
 member-configuration member sdb-rs3-s1-arbiter1
  host    10.192.2.34
  arbiter true
  site    remote
 exit
 member-configuration member sdb-rs3-s1-arbiter2
  host    10.192.2.44
  arbiter true
  site    remote
 exit
 member-configuration member sdb-rs3-s3-arbiter3
  host    10.192.2.22
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs3-s1-m1
  host     <Member_Host_IP>
  arbiter  false
  priority 104
  site     remote
 exit
 member-configuration member sdb-rs3-s1-m2
  host     10.192.2.33
  arbiter  false
  priority 103
  site     remote
 exit
 member-configuration member sdb-rs3-s2-m1
  host     10.192.2.42
  arbiter  false
  priority 102
  site     remote
```

```
 exit
 member-configuration member sdb-rs3-s2-m2
  host     10.192.2.43
  arbiter  false
  priority 101
  site     remote
 exit
exit
```

Note Configure a node label on all the master nodes.

Step 8 Validate the replica set status on cnAAA Ops Center using the `show db scdb replica-set-status` command.



Consistent replica set routing in apiRouter under high TPS

Feature History

Feature name	Release information	Description
Consistent replica set routing in apiRouter under high TPS	2025.03.0	This feature enables bulk provisioning for subscriber management within CPC system. It allows multiple create, read, update, and delete (CRUD) operations on number of subscribers through a single API call. This optimizes performance and enhances scalability when managing high transaction loads.

Overview

This feature enables bulk provisioning within the CPC system, allowing users to perform multiple subscriber management operations such as create, retrive, update, and delete (CRUD) through a single SOAP API call

through the apiRouter. It supports high transaction volumes, which improves efficiency and scalability by reducing the total number of API calls, lowering latency, and minimizing manual intervention.

How replica set routings in apiRouter under high TPS work

Perform bulk provisioning operations through the Api-Router in these stages:

- **Bulk API request initiation:** A request containing multiple subscriber operations is submitted.
- **apiRouter request processing:** The apiRouter receives and routes the bulk request.
- **Execution of bulk CRUD operations:** The system processes create, read, update, and delete actions for all specified subscribers.

Sample Bulk provision SOAP requests

Bulk provisioning SOAP requests allow the management of multiple subscriber records in a single operation. The following examples shows how to create, retrieve, update, and delete subscribers using the SOAP API. Each request targets a specific bulk operation and can include multiple subscriber entries.

Create Bulk Subscribers Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope
/" xmlns:typ="http://broadhop.com/unifiedapi/soap/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:CreateBulkSubscribersRequest>
         <subscriber>
            <name><fullName>BNGUser1</fullName></name>

<credential><networkId>0005.9A3C.7B1</networkId><password>abc123</password></credential>
            <service><code>A0F0100M100M000005MQ</code><enabled>true</enabled></service>
            <avp><code>CIRCLE_CODE</code><value>MU</value></avp>
            <status>ACTIVE</status>
         </subscriber>
         <subscriber>
            <name><fullName>BNGUser2</fullName></name>

<credential><networkId>0005.9A3C.7B2</networkId><password>abc123</password></credential>
            <service><code>A0F0100M100M000005MQ</code><enabled>true</enabled></service>
            <avp><code>CIRCLE_CODE</code><value>MU</value></avp>
            <status>ACTIVE</status>
         </subscriber>
      </typ:CreateBulkSubscribersRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Note Each `<subscriber>` element defines the details for one subscriber.

Get Subscribers Request

This request retrieves details for multiple subscribers by their network IDs.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ=
"http://broadhop.com/unifiedapi/soap/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:GetSubscribersRequest>
         <networkId>0005.9A3C.7B1</networkId>
         <networkId>0005.9A3C.7B2</networkId>
      </typ:GetSubscribersRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Update Subscribers Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope
/" xmlns:typ="http://broadhop.com/unifiedapi/soap/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:UpdateSubscribersRequest>
         <subscriber>
            <id>001000003e3ccd7f674d526e</id>
            <name><fullName>BNGUser1</fullName></name>
            <credential><networkId>0005.9A3C.7B1</networkId><password>
{SSHA}Z32udUfS0YsfTQj00KPjtZuJwPI/Pz9BSD8/Pw==</password></credential>
            <service><code>A0F0100M100M000030MQ</code><enabled>true</enabled>
</service>
            <status>ACTIVE</status>
            <avp><code>CIRCLE_CODE</code><value>MU</value></avp>
            <version>0</version>
         </subscriber>
         <subscriber>
            <id>002000003e3ccd7f674d52a4</id>
            <name><fullName>BNGUser2</fullName></name>
            <credential><networkId>0005.9A3C.7B2</networkId><password>
{SSHA}cnRbCtCz1RhaqQ+9JE+VPLheqt0
/P10/dz8/AQ==</password></credential>
            <service><code>A0F0100M100M000030MQ</code><enabled>true</enabled></service>
            <status>ACTIVE</status>
            <avp><code>CIRCLE_CODE</code><value>MU</value></avp>
            <version>0</version>
         </subscriber>
      </typ:UpdateSubscribersRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Note A Bulk Change of Authorization (CoA) request is generated only when the subscriber has active sessions.

Delete Subscribers Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope
/" xmlns:typ="http://broadhop.com/unifiedapi/soap/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:DeleteSubscribersRequest>
         <networkId>0005.9A3C.7B1</networkId>
         <networkId>0005.9A3C.7B2</networkId>
         <hardDelete>true</hardDelete>
```

```
      </typ:DeleteSubscribersRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Note If `<hardDelete>false</hardDelete>` is used or the tag is omitted, the subscriber record is not physically deleted from the database. Instead, the status is updated to 'deleted'.

Retrieve the KPIs for multiple subscriber CRUD operations

Procedure

To retrieve KPIs for multiple subscriber CRUD operations, follow these steps:

Step 1 Log into the Engine pod with this command:

```
kubectl exec -it engine-pod-name <namespace> bash
```

Step 2 Enter the curl command to get the KPIs related to subscribers:

```
curl -G http://127.0.0.1:8080/metrics | grep subscriber
```

Step 3 Review the output, which includes KPIs for:

a. Create multiple subscribers

```
action_total{node_type="unknown",type="create-bulk-subscribers",status="success",} 4.0
action_total{node_type="unknown",type="create-subscriber",status="error",} 1.0
action_total{node_type="unknown",type="create-subscriber",status="success",} 8.0
action_duration_seconds{node_type="unknown",type="create-bulk-subscribers",} 0.148
```

b. Read multiple subscribers

```
action_total{node_type="unknown",type="get-bulk-subscribers",status="success",} 3.0
action_total{node_type="unknown",type="get-bulk-subscribers",status="error",} 1.0
      action_duration_seconds{node_type="unknown",type="get-bulk-subscribers",}        0.052
```

c. Update multiple subscribers

```
action_total{node_type="unknown",type="update-bulk-subscribers",status="success",} 1.0
action_total{node_type="unknown",type="update-subscriber",status="error",} 1.0
action_total{node_type="unknown",type="update-subscriber",status="success",} 7.0
action_duration_seconds{node_type="unknown",type="update-subscriber",} 0.142
action_duration_seconds{node_type="unknown",type="update-bulk-subscribers",} 0.18
```

d. Delete multiple subscribers

```
action_total{node_type="unknown",type="delete-bulk-subscribers",status="success",} 5.0
action_total{node_type="unknown",type="delete-subscriber",status="success",} 8.0
action_duration_seconds{node_type="unknown",type="delete-bulk-subscribers",} 0.311
action_duration_seconds{node_type="unknown",type="delete-subscriber",} 0.12
```

Note Each KPI entry shows metrics such as total actions, status (success or error), and action duration in seconds.

Configurable MongoDB storage size

Feature Name	Release Information	Description
Configurable MongoDB storage size	2026.02.0	The MongoDB storage configuration feature enables you to define custom Persistent Volume Claim (PVC) storage sizes for individual MongoDB replica sets. This enhancement provides the flexibility to scale storage capacity for production deployments with large subscriber bases. This ensures efficient data management and prevents storage-related service interruptions.

The MongoDB storage configuration feature lets you define Persistent Volume (PV) storage sizes for individual MongoDB replica sets. It is useful in scenarios that require different data volumes, backup retention policies, or environment-specific settings. cnAAA previously enforced a hardcoded 5 Gi limit for all MongoDB Persistent Volume Claims (PVCs). You can change the default value to accommodate an increase in subscriber base or to meet specific data retention requirements.

Prerequisites and constraints:

- PVC storage size is immutable after creation. Changes to the storage configuration take effect only when the system is in a `system mode shutdown` state and the system recreates the PVCs during the subsequent restart.
- **Input validation**:
  - Minimum value: 5 Gi (to prevent MongoDB startup failures).
  - Maximum value: 250 Gi (to prevent excessive resource allocation).
  - Input format: Use integer values only (for example, 5, 10, or 100). cnAAA automatically appends the "Gi" suffix.

Configure MongoDB storage size

Use the Ops-Center CLI to configure the persistent volume storage size for MongoDB replica sets.

Procedure

Step 1 Ensure the system is in a shutdown state before you modify storage configurations.

```
system mode shutdown
```

Step 2 Enter the `config` mode.

Step 3 Specify the storage size (in Gi) for the desired replica set.

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

Example:

```
db scdb replica-name sdb-spr01 storage 10
```

Note Before configuring, you can view the range of storage sizes in the CLI help display.

```
[beta/beta-cncps] pcf(config)# db scdb replica-name sdb-spr01 storage ? Description:
Persistent volume storage size in Gi for this replica-set (e.g., 5, 10, 100). Range: 5-250.
 If not specified , uses default 5Gi Possible completions: <int, 5 .. 250>[100]
```

Step 4 `commit` the changes.

Step 5 Revert the system to the running state to trigger the recreation of the PVCs.

```
system mode running
```

Step 6 Verify storage configuration.

```
show running-config db scdb replica-name <replica-name>
```

Step 7 Check the capacity of the PVCs in the namespace:

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

```
# Enter configuration mode
config

# Configure sdb-spr01 with 10Gi of storage
and verify the pvc for the same
db scdb replica-name sdb-spr01 storage 10
kubectl get pvc -n pcf-beta-cncps | grep sdb-spr01
db-local-data-db-scdb-sdb-spr01-0       Bound
db-local-data-db-scdb-sdb-spr01-0-pcf-beta-cncps       10Gi        RWO
local-storage   <unset>                 63s

# Configure sdb-spr02 with 8Gi of storage
and verify the pvc for the same
db scdb replica-name sdb-spr02 storage 8
kubectl get pvc -n pcf-beta-cncps | grep sdb-spr02
db-local-data-db-scdb-sdb-spr02-0       Bound
db-local-data-db-scdb-sdb-spr02-0-pcf-beta-cncps      8Gi       RWO           local-storage
   <unset>                 63s
```

```
# Commit the changes
commit
```

Auto scheduler backup for MongoDB

Feature Name	Release Information	Description
Auto scheduler backup for MongoDB	2026.02.0	The automatic scheduler backup feature automates scheduled and on-demand MongoDB Subscriber Profile Repository (SPR) backups to remote Secure Copy Protocol (SCP) servers. This enhancement improves system reliability and reduces manual overhead by resolving authentication, and scheduling flexibility. Additionally, the system maintains a local safety net that retains the last three backup archives to ensure data resilience if external network transfers fail.

The automatic scheduler backup feature ensures high availability and recovery for the SPR by automating the backup of MongoDB data.

Administrators can configure periodic backup schedules or trigger on-demand backups. The system automatically transfers these backups to a remote SCP server while maintaining a local cache of the most recent backups to ensure data availability if the external network transfer fails.

Prerequisites

Before you configure the backup settings, ensure that you meet these requirements:

- **MongoDB version**: Use MongoDB 4.2 or higher.
- **Storage**: Ensure sufficient disk space is allocated to the Persistent Volume (PV). Allocate space equivalent to at least three times the total size of your active MongoDB dataset to support the three archives kept by the local round-robin safety net.
- **Security**: Ensure the `dbPassword` is AES-encrypted using the system's AES key.

Configure the MongoDB replica set

Use this procedure to configure the SPR replica sets.

Procedure

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

Step 2 Define the replica set members using their addresses.

Example:

```
db scdb replica-name sdb-spr01
  port 65001
  interface vlan2400
  resource cpu limit 3000
  resource memory limit 20000
  replica-set-label key smi.cisco.com/node-type
  replica-set-label value oam
  member-configuration member sdb-rs1-s1-m1
    host 2001:db8:1::32
    arbiter false
    priority 104
    site local
  exit
```

Step 3 Verify the replica set status after exiting the `config` mode.

```
show db scdb replica-set-status
```

Configure global backup settings

Use this procedure to configure the external SCP server, backup schedule, and database credentials.

Procedure

Step 1 Configure the SCP server details

```
db global-settings backup-settings scp-server host <scp_server_ip>
db global-settings backup-settings scp-server port 22
db global-settings backup-settings scp-server user-name <username>
db global-settings backup-settings scp-server password <encrypted_password>
db global-settings backup-settings scp-server remote-backup-path <path_on_scp_server>
```

Step 2 Set the frequency and time for automatic backups.

```
db global-settings backup-settings backup-frequency day <day>
db global-settings backup-settings backup-frequency time-to-backup <time>
```

Step 3 If Mongo authentication is enabled, configure these parameters.

```
db global-settings db-user-name <username>
db global-settings password <aes_encrypted_password>
```

Perform on-demand backups and monitoring

Use this procedure to trigger an immediate backup and monitor the status of the process.

1. Initiate an on-demand backup.

```
db scdb execute backup-data
```

2. Monitor the backup status.

```
show db scdb last-backup-status
```

Backup work-flow

The system automatically executes the backup and transfer work-flow through these sequential states. The system monitors for failure triggers at each transition point. If a failure occurs, the automation stops to prevent data corruption or incomplete transfers.

1. **Initialization**: The system starts the process in the `PENDING` state.
2. **Database identification**: The system identifies the database details. If the replica set is empty, or the identification fails, the system transitions to an error state or termination.
3. **Empty database handling**: If the system identifies an empty database within a replica set, it skips the replica set, logs the message "NO_DBS_TO_BACKUP", and proceeds to the next replica set.
4. **Data extraction**: Once the system determines the database details, it initiates the `DATA_DUMP_IN_PROGRESS` state. If this operation fails, the process halts.
5. **Archiving**: When the data dump completes successfully, the system transitions to the `ARCHIVING` state. If archive creation fails, the process halts.
6. **Secure transfer**: Occurs when the system creates the archive and triggers the `SCP_IN_PROGRESS` state to transfer the file. If the transfer fails, the process halts.
7. **Completion**: When the SCP transfer completes successfully, the system reaches the `SCP_DONE` state. This state concludes the automated workflow.

Restore the SCDB SPR backup

Use this procedure to manually restore the SPR data from a backup.

Procedure

Step 1 Run this command to extract the SCDB backup file.

```
tar -xvf <backup_filename>.tar.gz
```

Example:

```
cloud-user@beta-master-1:~$ tar -xvf mongodb_backup-2026-03-27-17-52-55.tar.gz
set04/spr/location_history.metadata.json
set04/spr/apirouter_sk_cache.metadata.json
set04/spr/auth_failures.metadata.json
set04/spr/prelude.json
set04/spr/location_history.bson
set04/spr/subscriber_ssid.metadata.json
set04/spr/subscriber.metadata.json
set04/spr/apirouter_sk_cache.bson
set04/spr/subscriber.bson
set04/spr/subscriber_ssid.bson
```

```
set04/spr/auth_failures.bson
cloud-user@beta-master-1:
```

Step 2 Transfer the extracted data into the `br-controller` pod. Use the `kubectl cp` command to move the extracted directory into the pod.

```
kubectl cp -n <namespace> <extracted_directory>/ <br-controller-pod-name>:/tmpkubectl cp -n <namespace>
 <extracted_directory>/ <br-controller-pod-name>:/tmp
```

Example:

```
cloud-user@beta-master-1:$ kubectl cp -n pcf-beta-cncps set04/ br-controller-scdb-556f6fd7c7-dlvcw:/tmp
```

Step 3 Execute the `mongorestore` command within the `br-controller` pod to restore the database:

```
kubectl exec -it -n <namespace> <br-controller-pod-name> -- bash
mongorestore --host <Primary-ip> --port <port> <extracted tar>
```

Example:

```
cloud-user@beta-master-1:~$ kubectl exec -it -n pcf-beta-cncps br-controller-scdb-556f6fd7c7-dlvcw
-- bash

I have no name!@br-controller-scdb-556f6fd7c7-dlvcw:/opt/workspace$ mongorestore --host 192.0.2.24
--port 65009 /tmp/set04/
.
.
.
.
2026-01-23T09:00:29.935+0000    4504095 document(s) restored successfully. 0 document(s) failed to
restore.
I have no name!@br-controller-scdb-556f6fd7c7-dlvcw:/opt/workspace$
```

Troubleshoot the local backup safety net

The system maintains a local safety net in the `/data/db/` directory of the `br-controller-scdb` pod.

- **FIFO retention**: The system retains the last three backup archives locally. If a fourth backup is created, the system automatically deletes the oldest archive.
- **Failure resilience**: If the system enters an `SCP_FAILED` state, it preserves the local archive. You can manually retrieve these files from the pod to ensure data recovery:

  ```
  kubectl exec -it <br-controller-pod> -- ls /data/db/
  ```

- **Authentication errors**: If the backup fails immediately, verify that the `dbPassword` is correctly AES-encrypted. Unencrypted passwords cause authentication failure during the dump process.

Subscriber authentication and Called-Station-ID uniqueness

The cnAAA features, subscriber authentication, and Called-Station-ID uniqueness validate the Called-Station-ID during authentication and ensure its uniqueness across Subscriber Profile Repository (SPR) shards.

Called-Station-ID uniqueness

In a multi-shard cnAAA deployment, subscribers are distributed across remote replica sets based on a hash of their Network-ID. Each shard enforces uniqueness within its own scope, however, it cannot detect if a Called-Station-ID also exists on another shard.

The Credential Index is a shared registry that records the shard that owns each Called-Station-ID. cnAAA checks this registry before it creates a subscriber. This check prevents two subscribers on different shards from using the same Called-Station-ID.

Called-Station-ID authentication

cnAAA validates the Called-Station-ID in a RADIUS Access-Request and also validates the Network-ID. Authentication succeeds only if the Network-ID matches the subscriber record and the Called-Station-ID authentication also succeeds. cnAAA returns an Access-Reject if the Called-Station-ID authentication fails.

For Air Fiber subscribers, cnAAA compares the A6 identifier and the last three octets of the MAC address and excludes the CSS portion. For FTTH subscribers identified by an OLT interface pattern, cnAAA compares the complete Called-Station-ID.

Deployment requirements for Called-Station-ID features

Refer to this topic to identify the requirements for deploying the Called-Station-ID features.

System requirements

- Use a Kubernetes cluster with sufficient nodes for high availability.
- Deploy MongoDB replica sets for each shard with geo-redundancy.
- Use a supported Java runtime environment for the API and service layers.
- Provide network connectivity between the API Router, Unified API, SPR, Policy Engine, and License Manager pods.
- Integrate monitoring and logging infrastructure.

Credential Index requirements

- Support subscriber create, read, update, and delete operations through the SOAP API with bulk provisioning.
- Maintain backward compatibility with single-database deployments.
- Support policy engine integration for dynamic policy updates when subscriber data changes.
- Support high availability and geo-redundancy in multi-shard environments.
- Use thread-safe operations with concurrency control.

Back-fill requirements

- Identify the primary host and port for each SPR replica set through Ops-Center.
- Ensure that all replica sets are healthy and that the system is fully operational.

Credential Index uniqueness enforcement

cnAAA uses reserve before create processing to prevent race conditions when subscribers are created on different SPR shards.

Summary

The key components in Called-Station-ID uniqueness enforcement are the Credential Index and the SPR shards.

- Credential Index: Stores a unique Called-Station-ID reservation and the owning shard.
- SPR shard: Stores the subscriber record after the reservation succeeds.

cnAAA reserves the Called-Station-ID in the Credential Index before it writes the subscriber record. A duplicate reservation prevents subscriber creation, and a failed subscriber write releases the reservation.

Workflow

These stages describe Called-Station-ID uniqueness enforcement:

1. A new subscriber creation request arrives at cnAAA.
2. cnAAA hashes the Called-Station-ID to locate the Credential Index shard and attempts to insert a reservation record.
3. If the Called-Station-ID is already registered, the Credential Index rejects the reservation and cnAAA returns a duplicate error without creating the subscriber.
4. If the reservation succeeds, cnAAA writes the subscriber data to the appropriate SPR shard.
5. If the subscriber write fails, cnAAA deletes the reservation from the Credential Index.

Credential Index records and configuration parameters

Refer to this topic to understand Credential Index records and configuration parameters.

Credential Index record

Each entry in the `credentialIndex` collection is a lightweight record:

```
{
  "_id"            : "HA_subscriber_2000001",
  "calledStationId": "IN001_2000001",
  "subscriberId"   : "0010000051f09e3866e19817",
  "ownerShardId"   : "shard-3",
  "createdAt"      : ISODate("2024-09-11T13:16:07.010Z")
}
```

Indexing and sharding

The unique index enforces Called-Station-ID uniqueness, and the hashed shard key distributes the collection across the existing replica sets.

```
// Ensure uniqueness for Called-Station-Id
db.credentialIndex.createIndex(
  { "calledStationId": 1 },
```

```
  { unique: true, name: "unique_calledStationId" }
);
```

Engine properties

Configure these properties in the engine properties in Ops Center:

Table 13: Credential Index engine properties

Parameter	Default	Description
`spr.credentialIndex.disabled`	ON	Set to `false` to enable global uniqueness checks.
`spr.credentialIndex.updateSubscriber.disabled`	ON	Set to `false` to synchronize the Credential Index during subscriber updates.
`spr.credentialIndex.getSubscriber.disabled`	ON	Set to `false` to enable cross-shard Called-Station-ID lookup.

Update and lookup behavior

The `updateSubscriber` property keeps the Credential Index synchronized when an update request changes or removes a subscriber Called-Station-ID. The old entry is released and the new entry is reserved.

The `getSubscriber` property enables cross-shard fallback. When a GetSubscriber request uses a Called-Station-ID as the `networkId` and the direct-shard lookup returns no record, cnAAA queries the Credential Index to locate the owning shard.

Configure global Called-Station-ID uniqueness

Use this procedure to enable Credential Index processing for Called-Station-ID uniqueness.

The Credential Index feature is disabled by default. Set each disabled property to `false` to enable its behavior.

Procedure

Step 1 Log in to PCF Ops-Center, then enter command mode.

Step 2 Enable global Called-Station-ID uniqueness checks.

```
engine <engine_name> properties spr.credentialIndex.disabled
  value false
exit
```

Step 3 Enable Credential Index synchronization for subscriber updates.

```
engine <engine_name> properties spr.credentialIndex.updateSubscriber.disabled
  value false
exit
```

Step 4 Enable cross-shard Called-Station-ID lookup.

```
engine <engine_name> properties spr.credentialIndex.getSubscriber.disabled
  value false
exit
```

Step 5 `commit` the changes and `exit`.

Example:

```
engine pcf-green properties spr.credentialIndex.disabled
  value true
 exit
 engine pcf-green properties spr.credentialIndex.getSubscriber.disabled
  value true
 exit
 engine pcf-green properties  spr.credentialIndex.updateSubscriber.disabled
  value true
 exit
```

Populate the Credential Index for existing subscribers

Use this procedure to populate the Credential Index for subscribers created before global MAC and Circuit ID uniqueness enforcement is enabled.

This one-time migration enables global MAC and Circuit ID (Called-Station-ID) uniqueness enforcement across the SPR databases. Complete the phases in order.

Before you begin

- Obtain access to Ops-Center. Confirm you have permissions to run `kubectl` commands, log in to a MongoDB or SCDB pod, and access an engine pod.
- Identify the PRIMARY host and port for each SPR replica set. Use these values in the migration commands.
- Ensure that all SPR replica sets are healthy and that the system is 100% operational.
- Schedule a maintenance window during which no CREATE or DELETE SOAP requests are expected. Updating Subscriber operations will not be affected during this period.
- Before deployment, verify that the `credentialIndex` collection does not exist on any SPR shard.

Procedure

Step 1 Identify the PRIMARY host and port for every SPR replica set.

a) List the database pods.

```
kubectl get pods -n <namespace> | grep scdb
```

b) Log in to Ops-Center.

c) Display the replica set status.

```
show db scdb replica-set-status | tab
```

d) Record the IP address and port of the node marked as PRIMARY for each SPR replica set.

Step 2 Complete the pre-migration checks.

a) Confirm that all SPR replica sets are healthy and that the system is 100% operational.

b) Confirm that no CREATE or DELETE SOAP requests are expected during the maintenance window.

Update Subscriber operations are not affected.

c) From a MongoDB or SCDB pod, check for the `credentialIndex` collection on all SPR shards.

```
for primary in <spr01-primary-ip>:<port> <spr02-primary-ip>:<port> <spr03-primary-ip>:<port>
<spr04-primary-ip>:<port>; do echo "Member==> $primary"; mongo $primary/spr --eval "show
collections"; done
```

d) Confirm that `credentialIndex` is absent from the output for every shard.

Step 3 Upgrade cnAAA to October 2026 release build.

a) Confirm that no startup errors occur after the upgrade.

b) Confirm that all engine, SCDB, and RADIUS endpoint pods are running and that the system is 100% operational.

Step 4 Verify the automatically created Credential Index collection and unique index.

The cnAAA engine pod automatically creates the unique index on the Called-Station-ID field of the `credentialIndex` collection at startup.

a) From a MongoDB or SCDB pod, confirm that `credentialIndex` exists on all SPR shards.

```
for primary in <spr01-primary-ip>:<port> <spr02-primary-ip>:<port> <spr03-primary-ip>:<port>
<spr04-primary-ip>:<port>; do echo "Member==> $primary"; mongo $primary/spr --eval "show
collections"; done
```

b) Verify that each shard has an index on Called-Station-ID with `"unique" : true`.

```
for primary in <spr01-primary-ip>:<port> <spr02-primary-ip>:<port> <spr03-primary-ip>:<port>
<spr04-primary-ip>:<port>; do echo "Member==> $primary"; mongo $primary/spr --eval
"printjson(db.credentialIndex.getIndexes())"; done
```

Step 5 Record the subscriber count for each SPR shard.

From a MongoDB or SCDB pod, run:

```
for primary in <spr01-primary-ip>:<port> <spr02-primary-ip>:<port> <spr03-primary-ip>:<port>
<spr04-primary-ip>:<port>; do echo "Member==> $primary"; mongo $primary/spr --eval
"db.subscriber.countDocuments()"; done
```

Step 6 Prepare the Credential Index Backfill Tool on an engine pod.

a) Locate an engine pod.

```
kubectl get pods -A | grep engine
```

b) Choose an engine pod and log in to it.

```
kubectl exec -it <pcf-engine-pod> -n <namespace> -- bash
```

c) Assemble the backfill JAR.

```
mkdir /tmp/credentialIndex_migration
cd /tmp/credentialIndex_migration

jar -xf /app/plugins/com.google.guava_*.jar
jar -xf /app/plugins/org.mongodb.mongo-java-driver_*.jar
jar -xf /app/plugins/com.broadhop.spr.migration.networkId.hashing_*.jar

jar -cfm credentialIndex-backfill.jar META-INF/MANIFEST.MF *
```

Step 7 Run the backfill tool in dry-run mode for each source SPR database.

Caution Every continued line must end with `\`. If a backslash is missing, bash treats the next argument as a separate shell command, and the Java tool does not receive the argument. The run can fail silently or return an unexpected error.

```
java -cp credentialIndex-backfill.jar \
  com.broadhop.spr.networkid.hashing.CredentialIndexBackfillTool \
  --subscriberHost <host>:<port> \
  --subscriberDbName <replica set db name> \
  --shard <replica set db name 1>=<host1>:<port1> \
  --shard <replica set db name 2>=<host2>:<port2> \
  --shard <replica set db name 3>=<host3>:<port3> \
  --shard <replica set db name 4>=<host4>:<port4> \
  --batchSize 500 \
  --dryRun \
  --logSkipped \
  --reportFile /tmp/credIdx_dry_dbspr1.log
```

Use these command options:

- `--subscriberHost`: Specifies the host and port of the source subscriber MongoDB shard to read.
- `--subscriberDbName`: Specifies the logical name of the source shard that is stored in the `shardId` field.
- `--shard`: Defines a target Credential Index shard. Repeat this option for every shard in the cluster by using the `name=host:port` format.
- `--targetShard`: Optionally writes only to the specified shard. Omit this option to write to all routed shards in one source-shard scan, as recommended by the MOP.
- `--batchSize`: Specifies the number of records in each bulk insert for a target shard. The default is 500.
- `--dryRun`: Scans and calculates routing without writing data.
- `--logSkipped`: Writes the subscriber `_id` and reason for each skipped subscriber to the report file.
- `--reportFile`: Specifies the output path for duplicate, error, and skipped records.

The dry run scans and reports without writing data.

Step 8 Validate the dry run and populate the Credential Index.

a) Review `/tmp/credIdx_dry_dbspr1.log`, confirm that its count matches the baseline count, and confirm that it contains no errors.

b) For each source SPR database, run the same command without the `--dryRun` option.

```
java -cp credentialIndex-backfill.jar \
  com.broadhop.spr.networkid.hashing.CredentialIndexBackfillTool \
  --subscriberHost <host>:<port> \
  --subscriberDbName <replica set db name> \
  --shard <replica set db name 1>=<host1>:<port1> \
  --shard <replica set db name 2>=<host2>:<port2> \
  --shard <replica set db name 3>=<host3>:<port3> \
  --shard <replica set db name 4>=<host4>:<port4> \
  --batchSize 500 \
  --logSkipped \
  --reportFile /tmp/credIdx_dry_dbspr1.log
```

Step 9 Verify the backfill counts on all SPR shards.

```
for primary in <spr01-primary-ip>:<port> <spr02-primary-ip>:<port> <spr03-primary-ip>:<port>
<spr04-primary-ip>:<port>; do echo "Member==> $primary"; mongo $primary/spr --eval
"db.credentialIndex.countDocuments()"; done
```

The MOP defines a successful backfill as a match between the Credential Index count and the subscriber count on every shard.

The Credential Index contains the backfilled entries, and its count matches the subscriber count as required by the MOP.

Disable global Called-Station-ID uniqueness

Use this procedure to disable global Called-Station-ID uniqueness.

Procedure

Step 1 Set `spr.credentialIndex.disabled` to `true` in Ops Center.

```
engine <engine_name> spr.credentialIndex.disabled
  value true
exit
```

The engine immediately stops using the Credential Index.

Step 2 (Optional) If you are permanently removing the feature, drop the `credentialIndex` collection on all shards.

```
mongo <primary-ip>:<port>/spr --eval "db.credentialIndex.drop()"
```

Called-Station-ID authentication rules

Refer to this topic to understand the Called-Station-ID authentication rules.

Authentication outcome

Authentication succeeds only when the existing Network-ID validation succeeds and Called-Station-ID validation succeeds. cnAAA returns an Access-Reject when Called-Station-ID validation fails, even when the Network-ID matches.

Air Fiber comparison

An Air Fiber Called-Station-ID uses the `CSS#A6/last-three-octets-of-MAC` format. For example:

```
INKABGLRAJNPNB0001ENBSLT001#INKABGLRAJNPNB0001PMPAA6002/FB:4C:B1
```

cnAAA extracts and compares the A6 identifier and the last three octets of the MAC address. It excludes the CSS portion because the CSS identifies a dynamic aggregation path that can change during subscriber mobility.

FTTH comparison

A Called-Station-ID that ends with an Optical Line Terminal (OLT) interface identifier is classified as a FTTH subscriber. For example:

```
INKLKOCHVYTLNB6001ENBOLT001#1/1/01/02/3
```

cnAAA compares the complete Called-Station-ID from the RADIUS Access-Request with the value in the SPR. The strings must match exactly.

Called-Station-ID authentication

Standard Policy Builder conditions support the complete string comparison used for FTTH subscribers. Air Fiber authentication requires a Java Action Phrase because it uses a partial-string comparison.

Summary

The key components in Called-Station-ID authentication are the RADIUS Access-Request, the SPR credential, the Java Action Phrase, and the policy engine.

- RADIUS Access-Request: Provides the request Called-Station-ID and Network-ID.
- SPR credential: Provides the provisioned Called-Station-ID for the subscriber.
- Java Action Phrase: Classifies, extracts, and compares Called-Station-ID values.
- Policy engine: Evaluates the stored comparison result and applies the authentication decision.

cnAAA performs the existing Network-ID authentication and compares the appropriate Called-Station-ID values. The policy flow returns an Access-Reject when the stored Called-Station-ID authentication result is false.

Workflow

These stages describe Called-Station-ID authentication:

1. cnAAA performs the existing Network-ID authentication.
2. The Java Action Phrase retrieves the Called-Station-ID from the SPR credential and the RADIUS Access-Request.
3. The action phrase classifies the subscriber as Air Fiber or FTTH based on the Called-Station-ID format.
4. For Air Fiber, the action phrase extracts the A6 and MAC-address suffix. For fiber, it extracts the complete Called-Station-ID.
5. The action phrase compares the extracted SPR value with the extracted Access-Request value.
6. Action phrase stores the authentication result and access type.
7. In the **Post Subscriber Load** rule flow, the policy engine evaluates the stored authentication result and returns an Access-Reject when the result is `false`.

Configure the Subscriber Authentication policy

Use this procedure to configure Policy Builder to invoke Called-Station-ID authentication.

The Java Action Phrase performs subscriber classification, value extraction, and comparison, and stores the result for the policy engine.

Procedure

Step 1 Login to cnAAA Ops-Center and Choose **Tools** > **Preferences**.

Step 2 Check **Show Policies editing options?**, select **Apply**, and select **Accept** on the warning message.

Step 3 Go to **POLICIES** >**Initial Blueprint**> **Post Subscriber Load**.

Step 4 Create the **`Subscriber Authentication`** policy structure.

a) Create a policy group named **`Subscriber Authentication`**.

b) Create a policy named **`Subscriber Authentication`** in the policy group.

Step 5 Add the condition **A RADIUS Access Request was received**.

Step 6 Add the condition **An SPR credential exists** and set **Type(String)** to **`Called-Station-Id`**.

Step 7 Add the condition **An IPolicyState exists**.

Step 8 Under **Actions**, add **Authenticate Subscriber Called-Station-Id**, set **Type** to **`Output`**, and map **RADIUS Request Called-Station-Id**, **SPR Credential**, and **PolicyState** to `Called Station Id`, `Credential`, and `IPolicyState`, respectively.

Step 9 Move the **Subscriber Authentication** policy group to the top of the **Post Subscriber Load** flow.

The **Post Subscriber Load** flow invokes Called-Station-ID authentication.

Configure Called-Station-ID rejection handling

Use this procedure to configure Policy Builder to reject a request when Called-Station-ID validation fails.

The policy engine evaluates the stored Called-Station-ID validation result and returns an Access-Reject when the result is `false`.

Procedure

Step 1 In the **Check Cred** policy, add the condition **An SPR Subscriber exists** before **An SPR Credential does not exist**.

Step 2 Open the **Authentication 1** policy and remove the condition **A SPR credential exists**.

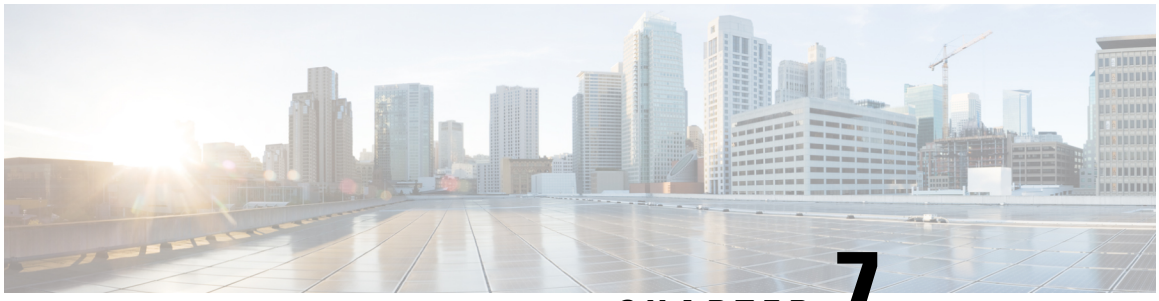
Step 3 Rename the policy to **`Called-Station-Id Authentication`**.

Step 4 Add the condition **An Authenticate Subscriber Called-station-Id Result exists** and set the **authenticated** parameter to **`false`**.

Step 5 Save and publish the Policy Builder configuration.

Step 6 Uncheck **Show Policies editing options?**, select **Apply**, and select **OK**.

The authentication policy rejects a request when the stored validation result is `false`.



CHAPTER 7

Subscriber migration from CPS 7.5 to cnAAA

- Subscriber migration from CPS 7.5 to cnAAA (Solution 1), on page 129
- Subscriber migration from CPS 7.5 to cnAAA (Solution 2) (SOAP Kafka Relay), on page 135

Subscriber migration from CPS 7.5 to cnAAA (Solution 1)

Feature History

Feature Name	Release Information	Description
Subscriber migration from CPS 7.5 to cnAAA	2025.04.0	This feature migrates subscriber management and policy enforcement from CPS 7.5 to the cnAAA platform to ensure service continuity during the transition. The migration is performed in phased, circle-by-circle upgrades with a SOAP proxy managing traffic and enabling temporary shared SPR access.

This feature migrates subscriber management and policy enforcement from CPS 7.5 to the cnAAA platform. It ensures service continuity during migration, intelligent API routing between platforms, and reliable rollback mechanism. It addresses key migration challenges, including staggered BNG transitions, single IP address constraints, and session data inconsistencies between platforms. A SOAP proxy manages traffic and allows temporary shared SPR access. This phased, circle-by-circle upgrade minimizes service disruption.

How subscriber migration works

This process describes how the system migrates subscriber data and sessions from CPS 7.5 to cnAAA.

1. **Initial traffic flow**: OCS and EMA send traffic directly to CPS 7.5 which manages subscriber data and sessions.
2. **Pre migration setup**:
    - The cnAAA component temporarily uses the SPR from Cisco Policy Suite 7.5.

- Reconfigure the OCS and EMA to point to the new SOAP Proxy, instead of directly to Cisco Policy Suite 7.5.

3. **API request handling by SOAP proxy during transition**:

   SOAP proxy intercepts all subscriber API requests (Create, Get, Update, Delete) from OCS and EMA and forwards them to CPS 7.5.

   - CPS 7.5 processes the **Create** or **Get** requests and sends a response to SOAP proxy which is then forwarded to both the OCS and the EMA. The cnAAA component is not involved in this process.
   - For **Update** and **Delete** requests, the SOAP proxy forwards them to CPS 7.5. If CPS 7.5 processes the request successfully, the SOAP proxy extracts the subscriber's MAC address and takes action:
     - For **Update** requests, it triggers a **Refresh Profile API** to cnAAA, which then processes any necessary CoA using the shared CPS 7.5 SPR and its CDL.
     - For **Delete** requests, it triggers a **Delete Session API** to cnAAA, which then removes the corresponding session from its CDL.

4. **Staggered BNG or circle migration**:

   The system gradually reconfigures BNGs within a circle to point to cnAAA. During this phase, some subscribers remain on CPS 7.5, while others move to cnAAA. The SOAP proxy routes requests for all subscribers.

5. **Circle completion**:

   Once all BNGs in a circle migrate to cnAAA, the OCS and EMA for that circle bypass the SOAP proxy and point directly to cnAAA. This completes the circle's transition.

6. **Overall Migration Completion**:

   After all circles migrate and point directly to cnAAA, the system exports the CPS 7.5 SPR database and imports it into cnAAA's local SPR. The system can then decommission CPS 7.5 for subscriber management.

HQoS Turbo Plan Rollout for Subscribers

This feature automates the migration of existing subscribers to new, higher-quality (HQoS) "Turbo Plans." It simplifies the process of upgrading customer services by enables internet service providers to apply enhanced bandwidth and QoS profiles across the subscriber base. This helps large-scale service upgrades and enhances the experience by minimizing manual effort.

Prerequisites:

Before executing the migration script, ensure the following prerequisites are met:

- **Policy Builder Configuration**: Add a new "Service Option" for the "Turbo Plan" to each HSI (High-Speed Internet) Service in Policy Builder for all relevant zones.
- **Execution Environment**: Run the HQoS_Migration_Script.py script from the master node of a setup with Persistent Volume (PV) enabled. Use master node-1 to execute the script.

Script execution

Procedure

To run the HQoS Migration Script, follow these steps:

Step 1 Create an input file named migrationData.csv which must contain comma-separated values with these columns:

- BNG_IP
- Service_Name

Example:

```
BNG_IP,Service_Name
192.168.101.98,TurboPlan_HSI_Service1
192.168.101.99,TurboPlan_HSI_Service2
```

Step 2 Go to the directory where the HQoS_Migration_Script.py file is located. and provide the full path to the script in the command.

Step 3 Run this command syntax to execute the script:

```
/data/pcf-m13/data-pcf-utilities-0/support/script/python
HQoS_Migration_Script.py --migrationfile=<path_to_migrationData.csv>
 --tps=<transactions_per_second> --port=<port_number> --soapurl=<unified_api_soap_url>
--opscentreip=<opscentre_ip_address> --pcfNamespace=<pcf_namespace>
```

How to verify PV enablement on the master node

Procedure

To verify Persistent Volume (PV) enablement on your Kubernetes master node, follow these steps:

Step 1 Access the Master Node and Verify `kubectl` Configuration.

Step 2 Enter the `kubectl get pv` command to list all PV configured in the cluster.

Note The presence of PVs in the output indicates that they are enabled. A `Bound` status confirms active allocation.

Step 3 Run the `kubectl get pvc -A` command to list all PV Claims across all namespaces.

Note PV Claims in a `Bound` state show that applications are successfully using storage. This means the PV is not only available but is also being used by workloads.

Step 4 Enter the `kubectl get sc` to verify the storage classes. **(Optional)**

Configure subscriber migration

Configure the SOAP Proxy, define API URLs, and establish database connectivity for the migration process.

Follow these steps to configure CPC migration:

Procedure

Step 1 Login to the Ops-Center and enter configuration mode.

```
config
```

Step 2 Enable the CPC migration and view available options.

a) **Enable migration**: `cpc-migration enable true`

b) **View options**: Use the `show full-configuration cpc-migration` command to display all available options and their descriptions.

```
cpc-migration enable [true | false]
```

Step 3 Configure the SOAP Proxy network access.

a) **Set external IP addresses**: `cpc-migration external-ips <IP1> <IP2>`

b) **Set the server port**: `cpc-migration properties server.port value <port value>`

Note Once configured, the SOAP Proxy can be accessed using the URL: `http://<configured_external_IP>:<configured_Server_Port>/soap`.

Step 4 Set the Unified API URLs for CPS 7.5 and cnAAA.

a) **CPS 7.5 URL**: `cpc-migration properties cps.url value <SOAP_URL_OF_CPS_7.5>/soap`

b) **cnAAA URL**: `cpc-migration properties cnaaa.url value <SOAP_URL_OF_cnAAA>/soap`

Example:

```
cpc-migration properties cps.url value https://192.0.2.184:8443/ua/soap
cpc-migration properties cnaaa.url value http://pcf-unified-api:8080/ua/soap/soap
```

Step 5 **Define SPR MongoDB connectivity**: Use the `cpc-migration ip-hostnames` command to associate IP addresses with their respective hostnames.

a) **Primary node**: `cpc-migration ip-hostnames <primary IP> hostname sessionmgr01-HA6`

b) **Secondary node**: `cpc-migration ip-hostnames <secondary IP> hostname sessionmgr02-HA6`

Step 6 Configure subscriber and replica properties.

a) **Enable GetSubscriber:** `cpc-migration properties enable.getsubscriber value true`

b) **Set replicas:** `cpc-migration replicas 3` (You can set a value from 1 to 5).

Step 7 (Optional) Configure debug logging.

a) **Set the logger port:** `cpc-migration logger-port <port number>`

b) **Set logging levels:**

```
cpc-migration debug logging logger com.cisco.cpc.migration.proxy
  level debug
exit
```

Step 8 Commit the changes and exit configuration mode.

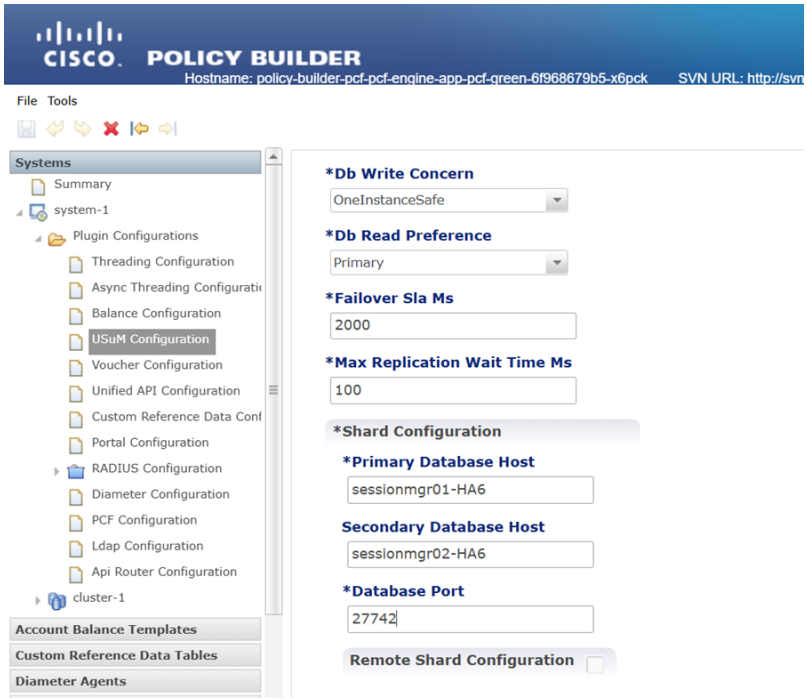
```
commit
exit
```

Note Once the system is up, configure the USum settings to connect to the CPS 7.5 SPR. This involves updating shared configurations with the primary database host, secondary database host, and database port of the CPS 7.5 SPR.

Note After configuring, delete the engine pods and control center pods this will restart them with the new configurations.

Policy Builder Configuration for USum

To connect cnAAA to the existing CPS 7.5 SPR database, the Policy Builder's USum settings must be configured.



Procedure

Step 1 In the Policy Builder GUI, navigate to **Plugin Configuration** > **USum configurations.**

Step 2 Enter the **Primary Database Host** configured in Ops Center.

Step 3 Enter the **Secondary Database Host** configured in Ops Center.

Step 4 Enter the **Database Port**.



Note To view CPS 7.5 SPR data in the cnAAA Control Center, remove the `-Dcc.ua.soap.url=<nil>` property from the Ops-Center configuration if it is present.

Logs available for migration

When subscriber migration is enabled from CPS 7.5 to the cnAAA, a dedicated `api-proxy-logging-0` Kubernetes pod deploys to capture and preserve logs from the SOAP API proxy pod. This ensures that log data, essential for debugging, monitoring, and tracking subscriber migration progress, persists even after SOAP API Proxy pod restarts.

These are the available log files for monitoring subscriber migration:

- **cnaaa-unfiedapi.log**: This file contains informational logs about requests sent to and responses received from the cnAAA Unified API, including Refresh Profile and Delete Session API calls.

  File path: `/data/api-proxy/cnaaa-unfiedapi.log`

  ```
  INFO log.cnAAA.unifiedAPI - Sending refresh_profile request <?xml version="1.0"
  encoding="UTF-8"?>
  <se:Envelope
  xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"><se:Body><RefreshSubscriberProfileRequest


  xmlns="http://broadhop.com/unifiedapi/soap/types"><networkId><![CDATA[0005.9A3C.7A00]]></networkId></RefreshSubscriberProfileRequest>
  </se:Body></se:Envelope> , from source IP Address: 192.168.64.64
  ```

- **cps-unfiedapi.log**: This file captures informational logs about requests sent to and responses received from the CPS 7.5 Unified API. It details the primary processing of subscriber operations (Create, Get, Update, Delete) by CPS 7.5.

  File path:`/data/api-proxy/cps-unfiedapi.log`

  ```
  INFO  log.cps.unifiedAPI - Received response: <se:Envelope
  xmlns:se="http://schemas.xmlsoap.org/soap/envelope/">
  <se:Body><CreateSubscriberResponse xmlns="http://broadhop.com/unifiedapi/soap/types">
  <errorCode>0</errorCode><errorMessage>Request completed
  successfully</errorMessage></CreateSubscriberResponse></se:Body>
  </se:Envelope> for Source IPAddress: 10.1.41.37
  ```

- **application.log**: This is a comprehensive application log file. It includes general operational information and specific details about the SOAP Proxy's processing logic and its interactions with CPS and cnAAA.

  File path: `/data/api-proxy/application.log`

```
INFO  c.c.cpc.migration.proxy.handler.SoapProxyHandler - Processing UPDATE_SUBSCRIBER
operation and status of
EnableGetSubscriber false
```

:

Subscriber migration from CPS 7.5 to cnAAA (Solution 2) (SOAP Kafka Relay)

Feature History

Feature Name	Release Information	Description
Subscriber migration from CPS 7.5 to cnAAA (SOAP Kafka Relay)	2026.01.0	The SOAP Kafka Relay solution for subscriber migration from CPS 7.5 to cnAAA, is an advanced solution which addresses the limitations of subscriber migration solution that uses the common CPS 7.5 SPR until the migration is completed. It addresses • performance and stability issues from shared database usage in CPS 7.5 and cnAAA, • introduces an asynchronous, Kafka-based replication mechanism to decouple database operations, and • ensures data consistency, system stability, and reduced operational risk during subscriber migration.

The SOAP Kafka Relay migrates subscriber data by functioning as a dual-write proxy. It synchronously services the primary legacy system (CPS 7.5) and asynchronously replicates changes to the new system (cnAAA). This decoupling prevents performance issues associated with shared database access.

Operational logic and monitoring

Core components

The architecture includes three primary components deployed within a Kubernetes environment:

- **SOAP proxy service (producer)**: Acts as the entry point for all Online Charging System (OCS) or Element Management System (EMA) SOAP traffic. It performs a synchronous write to CPS 7.5. If CPS

7.5 returns a success code (`errorCode 0`), the proxy publishes the request to the Kafka cluster and returns a success response to the client.

- **Kafka cluster:** Functions as a high-performance, durable buffer. It uses a partitioning strategy based on `networkId` to ensure that the system processes all messages for a specific subscriber in the exact order they were received. It also maintains a Dead Letter Queue (DLQ) for messages that fail replication after multiple attempts.
- **Consumer service:** Provides an asynchronous service that reads messages from the Kafka cluster. It executes the business logic required to update cnAAA and manages retries for downstream timeouts.

Data consistency

To ensure that cnAAA remains synchronized with CPS 7.5, the consumer service uses this process for updates:

- The service receives an `UpdateSubscriber` request.
- The service sends a `GetSubscriber` call to cnAAA.
- The service retrieves the current `<id>` and `<version>` from cnAAA.
- The service injects these values into the original update request and sends the request to cnAAA. This prevents version mismatches and ensures data integrity during the migration period.

High availability and Geo-Redundancy

The solution provides multi-site resilience using an active-passive model:

- MirrorMaker 2 (MM2): Replicates Kafka topics and consumer offsets between Site A and Site B.
- Heartbeat Mechanism: The passive site monitors heartbeats from the active site. If the active site becomes unavailable, the passive site detects the loss of heartbeats and activates its mirror or remote consumer group.
- Deterministic fail-over: Because offsets are replicated, the consumer at the secondary site identifies exactly where the primary site stopped. This ensures at-least-once delivery without data loss.
- Fail-back logic: When the primary site recovers, it compares local offsets with the processed offsets of the remote site to ensure it resumes from the most recent transaction.

Configure subscriber migration parameters in Ops-Center

Configure the SOAP Kafka Relay solution by performing this procedure. The parameters control dual-write behavior, Kafka messaging, rate limiting, and Geo-Redundancy settings.

Before you begin

Disable the previous migration solution before enabling the SOAP Kafka Relay.

```
set cpc-migration enable false
```

Procedure

Step 1 Log in to Ops-Center and enter the configuration mode.

```
config
```

Step 2 Configure core migration and server parameters.

a. Enable the SOAP Kafka Relay solution.

```
set cpc-migration soap-kafka-proxy enable true
```

b. Configure the SOAP proxy port and IP.

```
set cpc-migration soap-kafka-proxy port 8080
set cpc-migration soap-kafka-proxy external-ips [ <IP_Address> ]
```

Step 3 Configure pod affinity labels.

- `set cpc-migration soap-kafka-proxy affinity-label key <key_name>`

  `set cpc-migration soap-kafka-proxy affinity-label value <value_name>`

- Configure the label to the nodes where Kafka pods need to be deployed.

  `kubectl label node <node-name> <key_name>=<value_name>`

Step 4 Define primary and secondary URLs.

- The primary URL points to the active CPS 7.5 instance.
- The secondary URL points to the standby CPS 7.5 instance.

```
cpc-migration soap-kafka-proxy producer properties producer.primary.url
 value http://cps-7-5.example.com:8001/ua/soap
exit
cpc-migration soap-kafka-proxy consumer properties consumer.primary.url
 value http://pcf-unified-api:8080/ua/soap
exit
```

- Enable and configure the cnAAA secondary site URL for fail-over.

```
cpc-migration soap-kafka-proxy consumer properties secondary.url.enabled
 value true
exit

cpc-migration soap-kafka-proxy consumer properties consumer.secondary.url
 value http://pcf-unified-api:8080/ua/soap
exit
```

Step 5 Configure Geo-Redundancy

a. Enable cluster coordination for the consumer.

```
cpc-migration soap-kafka-proxy consumer properties cluster.coordination.enabled value <true|false>
```

b. Specify if the current site is the primary site.

```
cpc-migration soap-kafka-proxy consumer properties is.primary.site value <true|false>
```

c. Configure the local and remote Kafka site identifiers.

```
cpc-migration soap-kafka-proxy kafka local-site-id <Local_Site_Name>
cpc-migration soap-kafka-proxy kafka remote-site-id <Remote_Site_Name>
```

d. Configure the local Kafka external IP addresses.

```
cpc-migration soap-kafka-proxy kafka external-ip <local_Kafka_IP> <port>
exit
```

e. Configure the remote Kafka server IP addresses.

```
cpc-migration soap-kafka-proxy kafka remote-site kafka-server <remote_Kafka_IP> <port>
exit
```



Note Do not change the configured site-ID names. Changing these names creates separate Kafka topics for each ID, which impacts replication.

Step 6 Review the configuration to confirm that all URLs and site IDs are correct. After verifying, `commit` the changes.

```
show full-configuration cpc-migration soap-kafka-proxy
commit
```

Alerts

The system provides these alerts for monitoring and troubleshooting:

Critical Alerts

- Kafka consumer lag is consistently high or growing for more than 5 minutes.
- High rate of failures when sending requests to the primary or secondary systems.

Warning Alerts

High rate of messages sent to the DLQ.

DLQ processor script

The `subscriber_migration_dlq_processor.py` script retrieves subscriber messages from the `FailedApi-DLQ` Kafka topic. It then replays the contained SOAP requests against a specified consumer SOAP endpoint. Use this utility to recover or replay failed subscriber migration SOAP requests that are captured in the Dead Letter Queue (DLQ). This is helpful during troubleshooting or after you apply fixes to the consumer or service.

The script is located in the `/data/utilities/support/script/` directory within the `pcf-utilities` pod.

Run the DLQ processor

- Log in to the `pcf-utilities-0` pod from your master node:

  ```
  kubectl exec -it -n <namespace> pcf-utilities-0 -- bash
  ```

- Navigate to the script directory.

  ```
  cd /data/utilities/support/script/
  ```

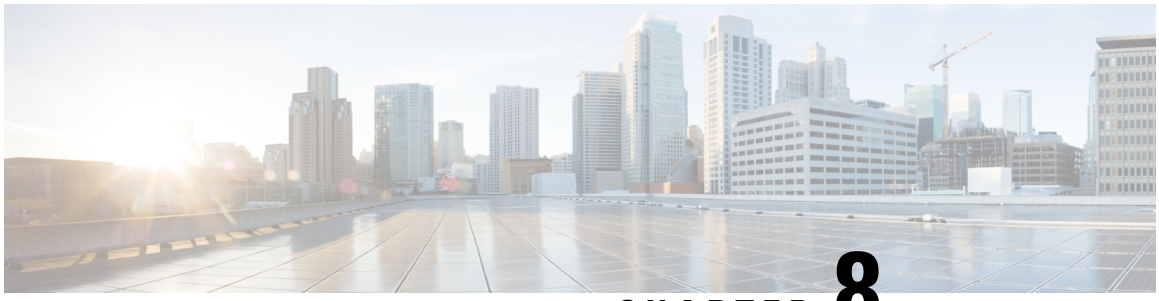
- Run the script without arguments and follow the on-screen prompts to specify the namespace, input type, value, and consumer URL.

  ```
  python3 subscriber_migration_dlq_processor.py
  ```

- Run the script with specific arguments.

Table 14: Command-line examples

Processing option	Command
Fetch the last 100 messages	`python3 subscriber_migration_dlq_processor.py 1 100 "https://unifiedapi/ua/soap" <namespace>`
Process from a specific Message ID	`python3 subscriber_migration_dlq_processor.py 2 <message-id> "https://unifiedapi/ua/soap" <namespace> --no-verify-ssl`
Process from a specific timestamp	`python3 subscriber_migration_dlq_processor.py 3 <timestamp> "https://unifiedapi/ua/soap" <namespace> --no-verify-ssl`



CHAPTER 8

RADIUS Endpoint

- Feature Summary and Revision History, on page 141
- RADIUS Endpoint, on page 144
- IPv6 Support, on page 150
- Cloud native CPS application on CNDP platform, on page 164
- Configure the RADIUS Endpoint in cnAAA using Ops-Center, on page 164
- Solution for Service Mismatch, on page 166
- Consolidation of CoA processing in a RADIUS endpoint for a given BNG, on page 179
- User Plane IP (vBNG) support, on page 183
- CoA throttling based on User Plane IP, on page 184
- Troubleshoot, on page 184
- Virtual NAS-IP label for CoA KPIs, on page 185
- CoA packet authenticator generation, on page 187
- Circle code labels for RADIUS KPIs, on page 190

Feature Summary and Revision History

Feature Description

Remote Authentication Dial-In User Service (RADIUS) attributes are used to define specific authentication, authorization, and accounting (AAA) elements in a user profile, which are stored on the RADIUS program.

Enable the RADIUS endpoint to dynamically create pods on a designated node or host. This feature is necessary to ensure nodes meet specific security and regulatory parameters are geographically closer to the datacenter. Node affinity determines the node where cnAAA creates the RADIUS endpoint pods, based on affinity towards a node or group of nodes. Node affinity is a set of rules that defines custom labels on nodes and specifies label selectors within the pods. Based on these rules, the scheduler determines the pod's placement location.

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

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

Overview

The CPC supports RADIUS for managing Authentication, Authorization, and Accounting (AAA) for users connecting to a network service. It ensures secure network access and precise session tracking for accurate billing and resource management. The feature also supports interim updates and Change of Authorization (CoA) requests, allowing real-time adjustments to user services. The Broadband Network Gateway (BNG) acts as the RADIUS client, sending requests to CPC based on user information.

The following are the core functionalities of the CPC in handling RADIUS requests:

- Access-Request
- Accounting-Request

Configure the node for the RADIUS Endpoint Pod

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



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

Mandatory RADIUS configuration

To configure the RADIUS server with essential parameters for optimal network performance and security, use the following configuration:

```
radius bind-ip <bind IP address>
radius replicas 2
radius settings request-timeout-ms 5000
radius settings max-tries  1
radius async-threading-configuration default-processing-threads 100
radius async-threading-configuration default-action-priority 5
radius async-threading-configuration default-action-threads 100
radius async-threading-configuration default-action-queue-size 400000
radius async-threading-configuration default-action-drop-oldest-when-full true
radius device-group ASR9K
  default-shared-secret     <secret value>
  default-coa-shared-secret <secret value>
  coa-port                  3799
 coa-timeout-seconds       3
  device <BNG Device Name>
   ip                 <BNG IP Address>
   shared-secret      <secret value>
   coa-shared-secret  <secret value>
   loopback-addresses [ <loopback addresses> ]
 exit
radius server-group <Server Group Name>
 servers <server name>
   primary              <primary OCS IP Address>
   secondary            <secondary OCS IP Address>
   nas-ip               <nas p address>
   accounting-port      1803
   authorization-port   1802
   auth-protocol        PAP
   radius-password      <radius password>
   shared-secret        <secret value>
```

```
   timeout-seconds      3
   test-message         false
   test-userid          test
   test-password        test123
   thread-pool-size     330
   max-proxy-queue-size 50000
       server-type   online (or) offline
   retries              0
 exit
radius properties grpc.executors
  value 5
exit
radius properties grpc.timeoutMs.processing
 value 5000
exit
radius properties io.netty.eventLoopThreads
  value 16
exit
radius properties parallelChannelCount
  value 5
exit
radius properties prometheusPort
  value 9099
exit
radius properties radiusCorePoolSize
  value 20
exit
radius properties radiusMaxQueue
  value 4000
exit
radius properties traps.tps
  value 4000
exit
radius properties udpMaxQueue
  value 4000
exit
radius properties udpPoolSize
 value 20
exit
```

Configure the RADIUS initialization delay

About the RADIUS initialization delay

The RADIUS initialization container can wait for old UDP conntrack and NAT entries to expire before the RADIUS pod starts. The delay range is 0 to 240 seconds. The default delay is 0 seconds, so the container starts without this wait.

Configure the delay

In Global Configuration mode, enter the delay in seconds. Commit the configuration. For example, to configure a 10-second delay, enter:

```
radius init-delay 10
commit
```

Verify the delay

Enter this command to verify the configured value:

```
do show running-config radius init-delay
```

To restore the default value, remove the command and commit the configuration:

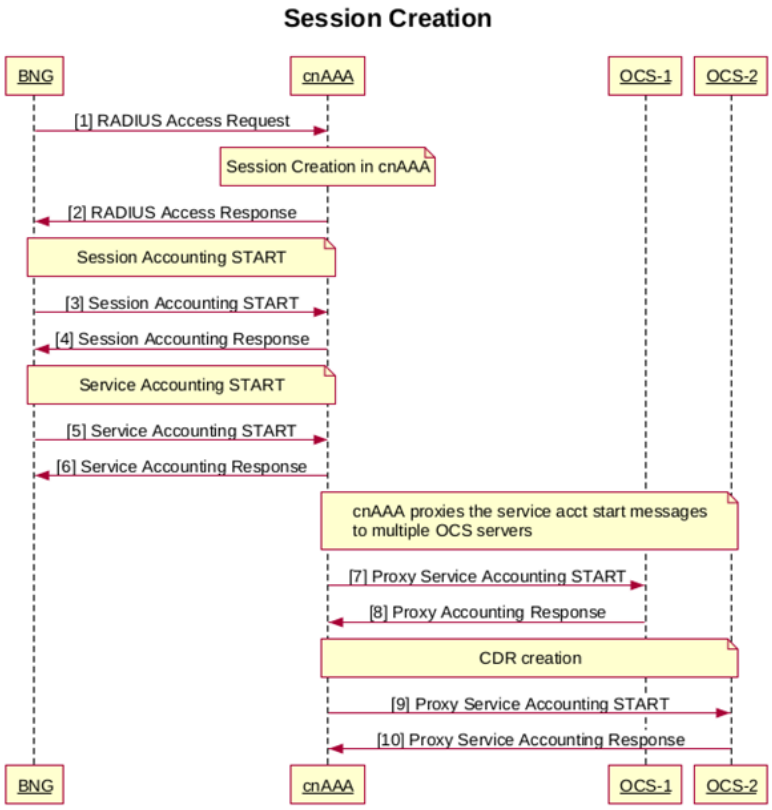
```
no radius init-delay
commit
```

RADIUS Endpoint

RADIUS call flow support in CPC

The RADIUS call flow in the CPC system outlines the steps involved in managing Authentication, Authorization, and Accounting (AAA) for network users. The process includes:

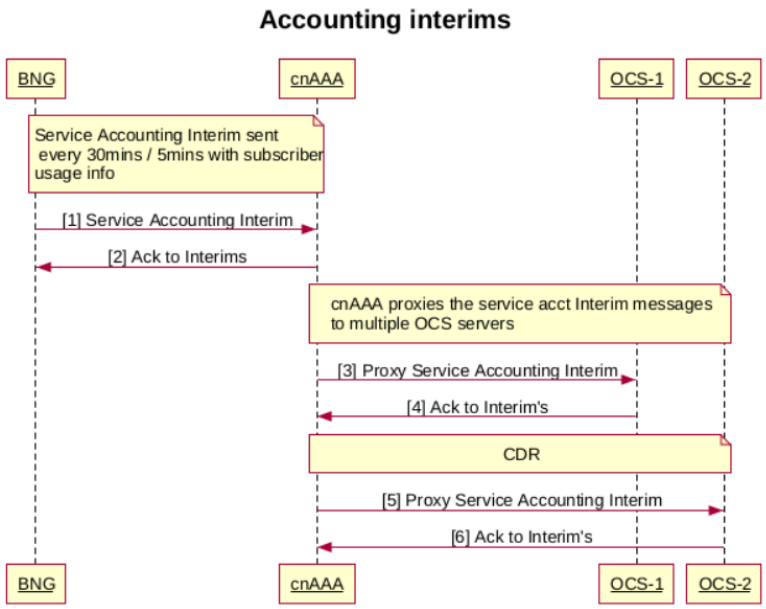
Session Creation



1	BNG sends RADIUS Access Request message to CPS
2	cnAAA sends Access-Response with the service name details
3	BNG Sends a session accounting START
4	cnAAA sends ACK to session accounting START
5	BNG sends a service accounting START to cnAAA
6	cnAAA sends ACK to service accounting START

7,8,9,10	cnAAA proxies the service accounting start to multiple OCS servers

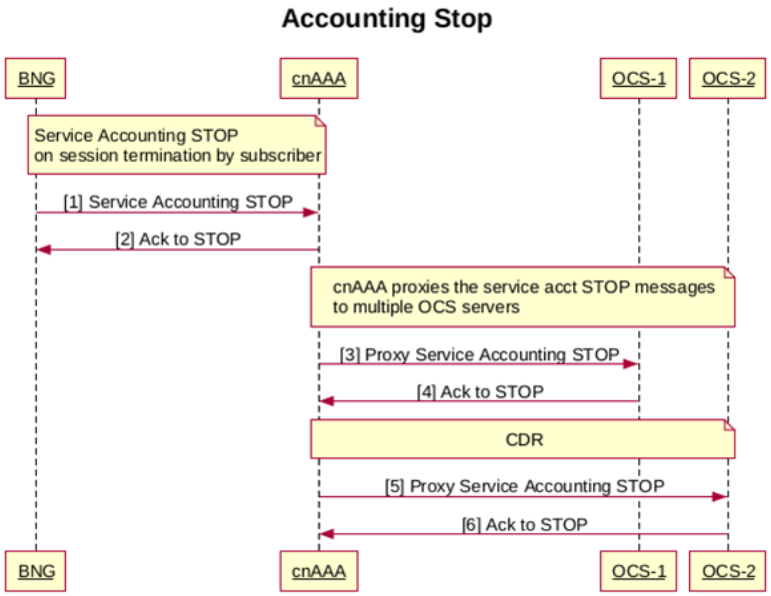
Accounting Interims Call Flow



1	BNG sends RADIUS Accounting Interim to cnAAA
2	cnAAA sends ACK to BNG
3,4,5,6	cnAAA proxies the service accounting start to multiple OCS servers

Accounting Stop Call Flow

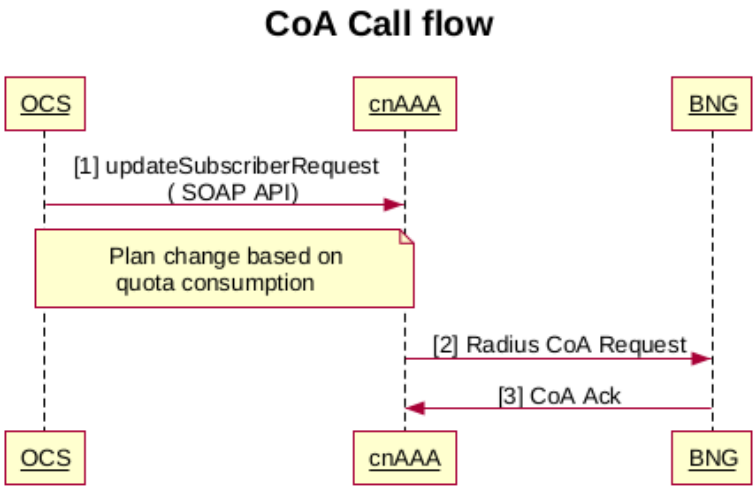
The Accounting Stop Call Flow in cnAAA includes the following steps:



1	BNG sends RADIUS Accounting STOP to cnAAA on session termination by subscriber
2	cnAAA sends ACK to BNG
3,4,5,6	cnAAA proxies the service accounting STOP to multiple OCS servers

Change of Authorization (CoA) Call Flow

The Change of Authorization (CoA) Call Flow in cnAAA involves the following steps:



1	OCS sends SOAP API request to cnAAA for a service change based on the quota consumption o
2	cnAAA updates the SPR and sends CoA to BNG to deactivate existing service and activate th name details
3	BNG sends Ack to CoA on successful service change for the subscriber

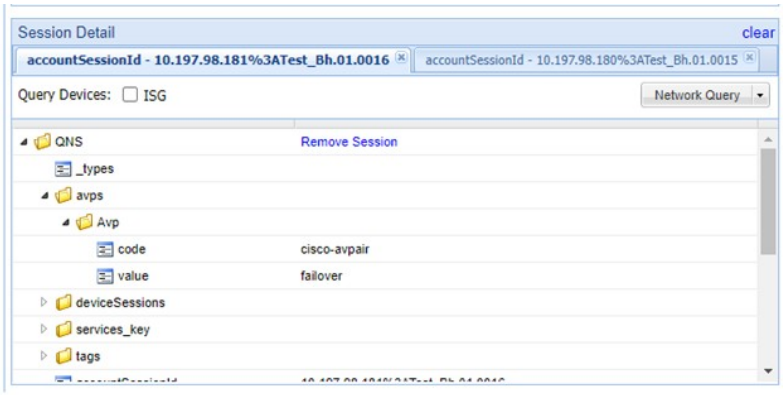
Handling CoA mismatch on BNG SRG switchover

During a BNG Service Redundancy Group (SRG) switchover, BNG2 establishes new sessions using services previously assigned on BNG1. CPS sends a CoA to BNG2 for any session that did not receive updated services on BNG1.

CPS is enhanced to resolve service mismatches resulting from the switchover. After the switchover, BNG2 sends two accounting start requests to create the new sessions:

- **Session accounting request**: This request contains the Attribute-Value Pair (AVP) `cisco-avpair=acct-trigger-cause=nas-switchover` and session ID `XXXX`.
- **Service accounting request**: This request contains the service details in the AVP `cisco-avpair=service-name=JIO_APPS`. The request includes session ID `YYYY` and `parent-session-id XXXX`.

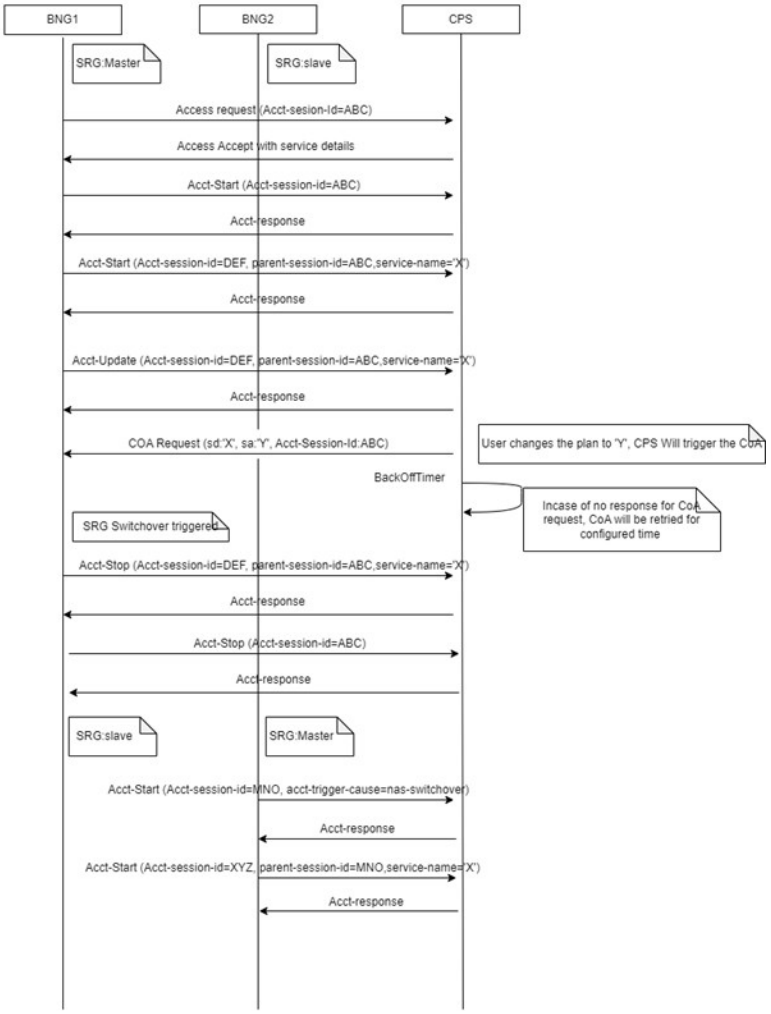
When CPS receives an accounting start request with `acct-trigger-cause` as `nas-switchover`, it creates a new session and marks it as a failover session.



When CPS receives a service accounting start request containing service details and a `parent-session-id`, it compares the user's service in CPS with the service in the request. If the services do not match, CPS triggers a CoA. If the services match, CPS sends an accounting response to the BNG.

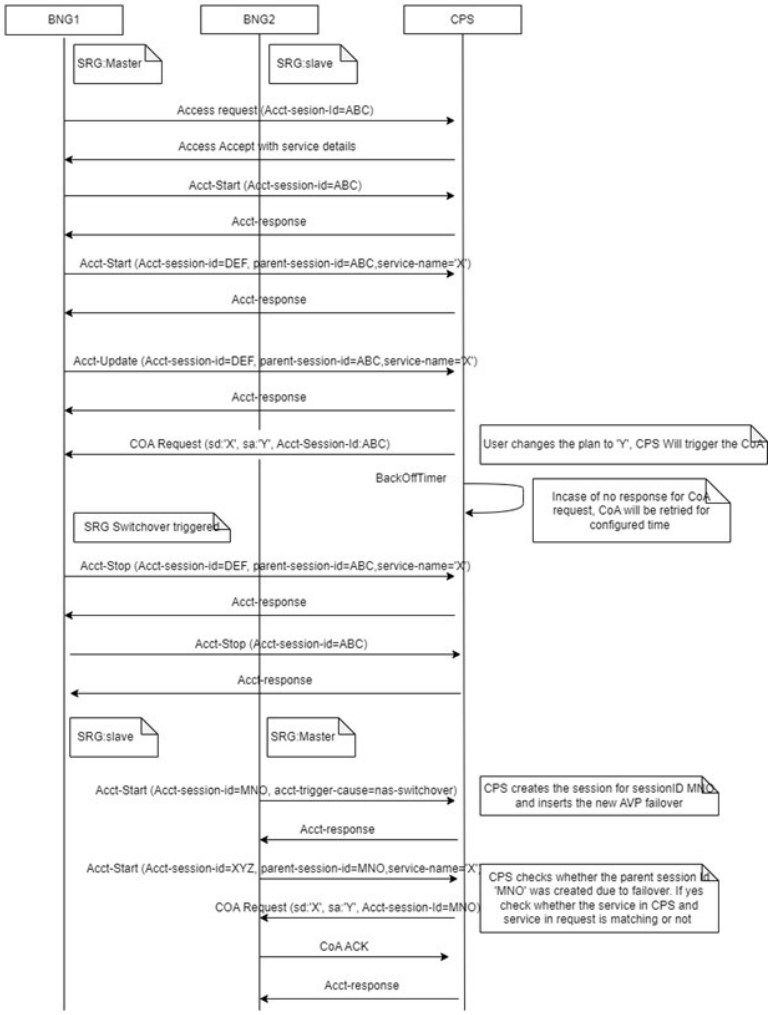
The existing behavior of BNG SRG switchover:

Existing Call flow of SRG Switchover scenario:



CPS has provided a solution to handle the service mismatch in BNG SRG switchover scenario:

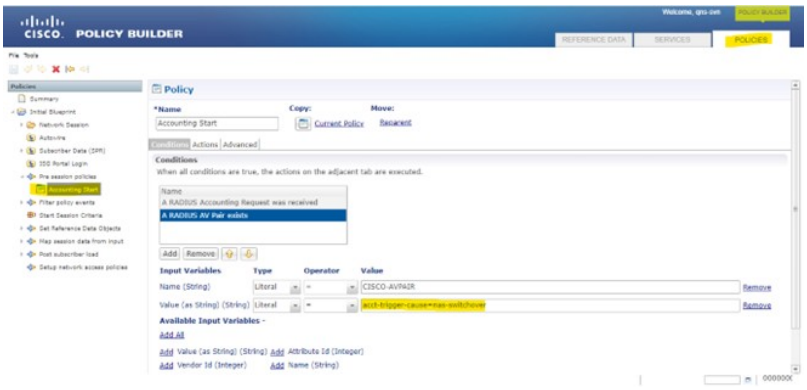
Call flow after handling the SRG Switchover scenario from CPS:



QNS Conf addition – Default Values if not given in qns.conf:

```
engine <engine group name> properties com.broadhop.SrgBngSwitchOverEnable value <true/false>
exit
```

Note Verify that the AVP name details are correct in the **Policies** section of **Policy Builder**.



IPv6 Support

Feature History

Overview

IPv6 support for cnAAA manages both IPv4 and IPv6 sessions, ensuring network connectivity.It operates in environments requiring either IPv4 or IPv6, supporting the transition to IPv6 while maintaining compatibility with existing IPv4 infrastructure without disrupting current operations.

To configure IPv6 support, configure the following components:

- Radius Endpoint
- Mongo Replica Sets

RADIUS Endpoint support in IPv6

RADIUS Endpoint support in IPv6 deploys cnAAA with IPv6/IPv4 dual stack connectivity to various network functions (NFs). In Ops-Center, the RADIUS endpoint configuration supports both IPv4 and IPv6 addresses for external IPs, device groups, and server group configurations. This setup ensures cnAAA operates effectively in different network environments, accommodating different IP address.

Properties to Enable IPv6

These properties enable the IPv6 support in cnAAA:

```
radius properties com.broadhop.pep.ipv6.enable.feature
value true
exit
engine cpc-green
policy-builder properties com.broadhop.pep.ipv6.enable.feature
value true
exit
```

Mongo Replica Sets configuration

Mongo replica set configurations support both IPv4 and IPv6 addresses for replica member hosts. This ensures that the initialization, replication, and initial sync functionalities of the Mongo replica set operate without any impact, regardless of whether IPv4 or IPv6 addresses are used. This dual support allows for flexible network configurations and aids in transitioning to IPv6 while maintaining compatibility with existing IPv4 infrastructure.

Each replica subscriber must be configured with either IPv4 or IPv6 addresses to ensure optimal functionality. It is not recommended to mix both IP versions within a single subscriber configuration.

Ops-Center configuration for enabling IPv6 in RADIUS Endpoint

To enable the IPv6 on the RADIUS Endpoint within CPC, complete the following configurations in Ops-Center.

General RADIUS settings

```
radius accounting-port 1813
radius authorization-port 1812
radius coa-port    1700
radius bind-ip 2001:ddff::1
radius settings request-timeout-ms 5000
radius settings max-tries  1
radius settings min-processing-time-millis 3000
radius settings backoff-time-millis 1000
```

BNG device group configurations

```
BNG Client Configuration
device bng01
  ip                 2002:20:50:53::100/127
  shared-secret      cisco
  coa-shared-secret  cisco
  loopback-addresses [ 12.0.0.1 ]
 exit
```

RADIUS server group configuration

```
radius server-group grp1
 servers DEL_OCS
  primary              2002:20:50:52::100
  secondary            2002:20:50:52::101
  accounting-port      1803
  authorization-port   1802
  auth-protocol        PAP
  radius-password      test123
  shared-secret        cisco
  timeout-seconds      5
  test-message         false
  test-userid          test
  test-password        test123
  thread-pool-size     400
  max-proxy-queue-size 40000
  server-type   online (or) offline
  retries              3
 exit
exit
```

Dual Stack Support

Dual Stack support for cnAAA manages of IPv4 and IPv6 sessions simultaneously, providing connectivity across networks. It also manages subsystem interfaces such as Policy Builder, Unified API, Control Center, and Central UI, and handles internal communications using IPv4 and IPv6.

Configure Dual Stack in cnAAA Ops Center

To configure Dual Stack in cnAAA Ops Center, follow these steps:

Procedure

Step 1 Log in to the Master Node.

Find Ops Center IP: Use `kubectl get svc -n pcf | grep ops` command to identify the IP for secure access.

Connect through SSH: Use `ssh -p 2024 admin@<Ops Center IP>` to log in to the Ops Center.

Step 2 Enter `show running-config` to verify the current setup.

Step 3 Enter the configuration mode using the `config` command.

Step 4 Set the IPv4 and IPv6 parameters and enter the `commit` to commit and save the configuration.

Step 5 Enter the `show full-configuration radius` command to verify that the RADIUS configuration has been correctly implemented and saved.

```
radius accounting-port 1812
radius authorization-port 1813
radius coa-port    2799
radius bind-ipv4   [ 192.0.2.1, 198.51.100.1 ]
radius bind-ipv6   [ 2001:DB8::1, 2001:DB8:1::1 ]
radius replicas    3
radius lbs-service true
radius settings request-timeout-ms 5000
radius settings max-tries  4
radius settings min-processing-time-millis 3000
radius settings backoff-time-millis 1000
radius advance-tuning port-range-limit 699
radius device-group ASR9K
 default-shared-secret     $8$qPdz5EmrhWJB0SNpzVz54be7YDK4mbRrLjx9rb+FDYc=
 default-coa-shared-secret $8$3fKlmnu0rPehxoRYKZVpDuLBrHopO5jNBptAQAmqjYE=
 device bng1
  ip                 192.0.2.1
  shared-secret      $8$7jPNi/1n9d5TPrvmooTvueNqensExCS3aqaPc919R/U=
  coa-shared-secret  $8$txf4uvoj0VPF/L0jd4yt/ta+j53wbzWJtpqEa8ht03c=
  loopback-addresses [ 12.3.1.2 ]
 exit
 device bng2
  ip                 2001:DB8::1
  shared-secret      $8$Lk7h3H3k9z+xPVTwo7eSCNRf9BQGwP7dV+V3CA3/9fw=
  coa-shared-secret  $8$3aqauvoj0VPF/L0jd4yt/ta+j53wbzWJtpqEa8ht03c=
  loopback-addresses [ 12.3.1.2 ]
 exit
exit
radius server-group grp1
 servers serv1
  primary              2001:DB8:1:1::1
  secondary            2001:DB8:1:2::1
```

```
  nas-ip               2001:DB8:1:3::1
  accounting-port      8312
  authorization-port   1312
  auth-protocol        PAP
  radius-password       $8$nbssKHR2U1zrXCZQSXUB0dfxANe4R5c5+AGFHC1ZAnk=
  shared-secret        $8$fUCcJ4PFvLIbMWmkeqvfAEpWeKYE419RqR+48hHRJ64=
  timeout-seconds      20
  test-message         false
  test-userid          test
  test-password        $8$hqZmdZSkCs2kLuygYVZMv9YSXLO4OFwVphTQEZQt8Lw=
  thread-pool-size     10
  max-proxy-queue-size 3
 exit
exit
```

Note The application decrypts the password for internal use.

Ops Center configuration for enabling an external SNMP server

To enable an external SNMP server within CPC, following configurations in Ops Center are required:

Procedure

Step 1 Log in to the Ops Center.

Step 2 Enable and Configure External SNMP Server.

- Enable `snmp-trapper enable` to `true` to activate SNMP trapping.
- Configure the `snmp-trapper v2c-target` with the desired IPv4 or IPv6 address of the external SNMP server.
- Set the `port` to `162` for SNMP communication.
- Configure the `community` with the string `Re4D0nLy5TrinG` for SNMP access.
- Enter the `exit` command to complete and exit the configuration mode.

Overview

Prerequisite:

Ensure that the deployment environment is configured for dual-stack operation (both IPv4 and IPv6 enabled). For detailed instructions on how to set up a dual-stack environment, please refer to the Configure Cluster Manager for Dual Stack Environment.

This feature introduces IPv6 support across cnAAA to address deployment limitations in dual-stack network environments. It extends existing IPv4 functionalities to IPv6 and enables dual-stack operations for critical components. This includes IPv6 for CPC subscriber provisioning, SNMP server integration, dual-stack for

the Unified API and SNMP server, internal communication between CPC services, and Geo-Redundancy (GR) support for the CDL application.

Configure cnAAA Ops-Center for IPv6

Follow these steps to configure cnAAA for a dual stack environment.

Procedure

Step 1 Login to Ops Center and enter the configuration mode.

```
config
```

Step 2 Bind the RADIUS service to an IPv6 address.

```
radius bind-ipv4 [ 192.0.2.200 ]
radius bind-ipv6 [ 2001:DB8:20:50:51::200 ]
```

- Configure RADIUS device groups with IPv6 addresses for network devices.

  Example:

  ```
  radius device-group ASR9K
  device v6bng01
  ip 2001:DB8:20:50:53::100/125
  shared-secret $8$...[HASHED_PASSWORD]
  coa-shared-secret $8$...[HASHED_PASSWORD]
  loopback-addresses [ 127.0.0.1 ]
  exit
  exit
  ```

- Configure RADIUS server groups with IPv6 addresses for primary and secondary servers.

  Example:

  ```
  radius server-group grp1
   servers DEL_OCS
    primary              2001:DB8:20:50:59::100
    secondary            2001:DB8:20:50:59::101
    accounting-port      1803
    authorization-port   1802
    auth-protocol        PAP
    radius-password      $8$bYXTCjptJ6ILhivC4SjQHdf3+Wmpu9klu3qXcBnR2ck=
    shared-secret        $8$vVcdZ5CCnPAK2Wbl8FTjqGiqss5ls5LrVF9Sl1460PI=
    timeout-seconds      5
    test-message         false
    test-userid          test
    test-password        $8$M4odrqCNL+WjJcs11Yf5r+xTT637JDLm7zxDcJLrBe4=
    thread-pool-size     300
    max-proxy-queue-size 30000
    retries              3
   exit
   # ... (Include other relevant OCS servers as needed)
  exit
  ```

- Enable the IPv6 feature for RADIUS Policy Enforcement Point (PEP).

  ```
  radius properties com.broadhop.pep.ipv6.enable.feature
  value true
  exit
  ```

Step 3 Configure SCDB replica sets on IPv6 for each member.

Example:

```
db scdb replica-name admin-db
 port      65008
 interface vlan2010
 resource cpu limit 2000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs1-s1-m1
  host     2001:DB8:20:50:60::22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs1-s1-m2
  host     2001:DB8:20:50:60::23
  arbiter  false
  priority 101
  site     local
 exit
 member-configuration member sdb-rs4-arbiter
  host    2001:DB8:20:50:60::24
  arbiter true
  site    local
 exit
exit
db scdb replica-name spr1
 port      65001
 interface vlan2010
 resource cpu limit 4000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs1-arbiter
  host    2001:DB8:20:50:60::24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs1-s1-m1
  host     2001:DB8:20:50:60::22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs1-s1-m2
  host     2001:DB8:20:50:60::23
  arbiter  false
  priority 101
  site     local
 exit
exit
db scdb replica-name spr2
 port      65002
 interface vlan2010
 resource cpu limit 4000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs2-arbiter
  host    2001:DB8:20:50:60::24
  arbiter true
  site    local
```

```
 exit
 member-configuration member sdb-rs2-s1-m1
  host     2001:DB8:20:50:60::22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs2-s1-m2
  host     2001:DB8:20:50:60::23
  arbiter  false
  priority 101
  site     local
 exit
exit
db scdb replica-name spr3
 port      65003
 interface vlan2010
 resource cpu limit 4000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs3-arbiter
  host    2001:DB8:20:50:60::24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs3-s1-m1
  host     2001:DB8:20:50:60::22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs3-s1-m2
  host     2001:DB8:20:50:60::23
  arbiter  false
  priority 101
  site     local
 exit
exit
db scdb replica-name spr4
 port      65004
 interface vlan2010
 resource cpu limit 4000
 resource memory limit 60000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value oam
 member-configuration member sdb-rs4-arbiter
  host    2001:DB8:20:50:60::24
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs4-s1-m1
  host     2001:DB8:20:50:60::22
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs4-s1-m2
  host     2001:DB8:20:50:60::23
  arbiter  false
  priority 101
  site     local
 exit
exit
```

Step 4 Enable IPv6 features within the engine configuration.

Example:

```
api unified engine-group production-rjio
api unified externalIPs [ 192.0.2.87 2001:DB8:1::87 ]
engine production-rjio
properties com.broadhop.domain.ipv6.enable.feature
value true
exit
properties com.broadhop.pep.ipv6.enable.feature
value true
exit
policy-builder properties com.broadhop.pep.ipv6.enable.feature
value true
exit
crdapi admin-db primary 2002:20:50:60::22
crdapi admin-db secondary 2002:20:50:60::23
crdapi admin-db port 65008
```

Configure Geo-redundancy with CDL over IPv6

The CDL application supports GR over IPv6, enabling replication between sites using IPv6 communication. Follow these steps to configure GR with CDL over IPv6.

Procedure

Step 1 Login to CPC Ops-Center and enter the configuration mode.

```
config
```

Step 2 Configure the remote site's database endpoint and Kafka servers with IPv6 addresses.

Example:

```
cdl remote-site 2
db-endpoint host 2002:10:50:58::201
db-endpoint port 8882
kafka-server 2002:10:50:58::201 10091
exit
kafka-server 2002:10:50:58::201 10092
exit
kafka-server 2002:10:50:58::201 10093
exit
exit
```

Step 3 Configure the local CDL datastore session endpoint and Kafka external IP addresses with IPv6 addresses.

Example:

```
cdl datastore session
endpoint external-ipv6 2002:10:50:58::200
exit
cdl kafka external-ipv6 2002:10:50:58::200 10091
exit
```

```
cdl kafka external-ipv6 2002:10:50:58::200 10092
exit
cdl kafka external-ipv6 2002:10:50:58::200 10093
exit
```

Step 4 Verify Geo-synchronization status using the `verify_geo_sync` command on CDL endpoint pods.

Example:

```
cloud-user@gamma-master-1:/data/pcf-gamma-cncps/data-pcf-utilities-0/support/script$
NAMESPACE="pcf-gamma-cncps" REMOTE_IP="2002:10:50:58::201" IPFAMILY=6 ./initiate_geo_sync.sh
Defaulted container "ngn-datastore-ep" out of: ngn-datastore-ep, k8tz (init)
2026/06/16 16:51:02 Geo sync is successful
Defaulted container "ngn-datastore-ep" out of: ngn-datastore-ep, k8tz (init)
2026/06/16 16:51:04 Geo sync is successful
cloud-user@gamma-master-1:/data/pcf-gamma-cncps/data-pcf-utilities-0/support/script$
```

Step 5 Check session counts replication on each site.

Example:

```
[gamma/gamma-cncps] pcf# cdl show sessions count summary
Wed Sep  24 07:37:24.899 UTC+00:00
message params: {cmd:session-count-summary mode:cli dbName:session sessionIn:{mapId:0 limit:500 key:
 purgeOnEval:0 filters:[] inFilters:[] nextEvalTsStart:0 nextEvalTsEnd:0 createTsStart:0 createTsEnd:0
 lastUpdateTsStart:0 lastUpdateTsEnd:0 allReplicas:false maxDataSize:4096 mapIDs:[] sliceNames:[]
rebalancePublishTPS:0 createdBefore:0 concurrencyFactor:0 file:} sliceName:}
count 4202070
site-session-count {
    system-id 1
    count 2101032
}
site-session-count {
    system-id 2
    count 2101038
}
[gamma/gamma-cncps] pcf#
```

```
[delta/delta-cncps] pcf# cdl show sessions count summary
Wed Sep  24 07:37:28.328 UTC+00:00
message params: {cmd:session-count-summary mode:cli dbName:session sessionIn:{mapId:0 limit:500 key:
 purgeOnEval:0 filters:[] inFilters:[] nextEvalTsStart:0 nextEvalTsEnd:0 createTsStart:0 createTsEnd:0
 lastUpdateTsStart:0 lastUpdateTsEnd:0 allReplicas:false maxDataSize:4096 mapIDs:[] sliceNames:[]
rebalancePublishTPS:0 createdBefore:0 concurrencyFactor:0 file:} sliceName:}
count 4202098
site-session-count {
    system-id 1
    count 2101045
}
site-session-count {
    system-id 2
    count 2101053
}
```

Configure Geo-redundancy with Mongo for SPR or Admin DB over IPv6

The Mongo supports GR over IPv6, enabling replication between sites using IPv6 communication. Follow these steps to configure GR with SPR or Admin DB over IPv6.

Procedure

Step 1 Login to CPC Ops-Center and enter the configuration mode.

```
config
```

Step 2 Configure the Mongo replica set across GR cluster and third site arbiter with IPv6 addresses.

Example:

```
GR Site1: SPR

db scdb replica-name sdb-spr01
 port      65001
 interface vlan2400
 resource cpu limit 3000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value spr
 member-configuration member sdb-rs1-s1-arbiter1
  host    2001:DB8:1::34
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs1-s1-arbiter2
  host    2001:DB8:1::44
  arbiter true
  site    remote
 exit
 member-configuration member sdb-rs1-s1-m1
  host     2001:DB8:1::32
  arbiter  false
  priority 104
  site     local
 exit
 member-configuration member sdb-rs1-s1-m2
  host     2001:DB8:1::33
  arbiter  false
  priority 103
  site     local
 exit
 member-configuration member sdb-rs1-s2-m1
  host     2001:DB8:1::42
  arbiter  false
  priority 102
  site     remote
 exit
 member-configuration member sdb-rs1-s2-m2
  host     2001:DB8:1::43
  arbiter  false
  priority 101
  site     remote
 exit
 member-configuration member sdb-rs1-s3-arbiter3
  host    2001:DB8:1::22
  arbiter true
  site    remote
 exit
exit
```

```
GR Site2: SPR

db scdb replica-name sdb-spr01
 port      65001
 interface vlan2400
 resource cpu limit 3000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value spr
 member-configuration member sdb-rs1-s1-arbiter1
  host    2001:DB8:1::34
  arbiter true
  site    remote
 exit
 member-configuration member sdb-rs1-s1-arbiter2
  host    2001:DB8:1::44
  arbiter true
  site    local
 exit
 member-configuration member sdb-rs1-s1-m1
  host     2001:DB8:1::32
  arbiter  false
  priority 104
  site     remote
 exit
 member-configuration member sdb-rs1-s1-m2
  host     2001:DB8:1::33
  arbiter  false
  priority 103
  site     remote
 exit
 member-configuration member sdb-rs1-s2-m1
  host     2001:DB8:1::42
  arbiter  false
  priority 102
  site     local
 exit
 member-configuration member sdb-rs1-s2-m2
  host     2001:DB8:1::43
  arbiter  false
  priority 101
  site     local
 exit
 member-configuration member sdb-rs1-s3-arbiter3
  host    2001:DB8:1::22
  arbiter true
  site    remote
 exit
exit


Third Site Arbiter:

db scdb replica-name sdb-spr01
 port      65001
 interface vlan2400
 resource cpu limit 3000
 resource memory limit 20000
 replica-set-label key smi.cisco.com/node-type
 replica-set-label value spr
 member-configuration member sdb-rs1-s1-arbiter1
  host    2001:DB8:1::34
  arbiter true
```

```
  site    remote
 exit
 member-configuration member sdb-rs1-s1-arbiter2
  host    2001:DB8:1::44
  arbiter true
  site    remote
 exit
 member-configuration member sdb-rs1-s1-m1
  host     2001:DB8:1::32
  arbiter  false
  priority 104
  site     remote
 exit
 member-configuration member sdb-rs1-s1-m2
  host     2001:DB8:1::33
  arbiter  false
  priority 103
  site     remote
 exit
 member-configuration member sdb-rs1-s2-m1
  host     2001:DB8:1::42
  arbiter  false
  priority 102
  site     remote
 exit
 member-configuration member sdb-rs1-s2-m2
  host     2001:DB8:1::43
  arbiter  false
  priority 101
  site     remote
 exit
 member-configuration member sdb-rs1-s3-arbiter3
  host    2001:DB8:1::22
  arbiter true
  site    local
 exit
exit
```

Step 3 Verify the replica set status from any of the cluster (site 1, site 2 or third site arbiter)

```
show db scdb replica-set-status
```

Sample output:

```
db scdb replica-set-status sdb-spr01
```

```
=============================================================================================================================
HostName          Port      MemberName          NodeName
     Priority                  State                    IsArbiter           Replication-lag   Site
  (IP)
Running    Config   From-Primary   From-Member    Running   Config           (Seconds)
=============================================================================================================================
2001:DB8:1:2::32  65001     sdb-rs1-s1-m1       gamma-master-1
104        104      PRIMARY        PRIMARY        false     false        0.0               local
2001:DB8:1:2::22  65001     sdb-rs1-s3-arbiter3 beta-master-1
0                    ARBITER        ARBITER        true      true         N/A               remote
2001:DB8:1:2::44  65001     sdb-rs1-s1-arbiter2 delta-master-3
0                    ARBITER        ARBITER        true      true         N/A               remote
2001:DB8:1:2::43  65001     sdb-rs1-s2-m2       delta-master-2
101        101      SECONDARY      SECONDARY      false     false        0.0               remote
2001:DB8:1:2::33  65001     sdb-rs1-s1-m2       gamma-master-2
103        103      SECONDARY      SECONDARY      false     false        0.0               local
2001:DB8:1:2::34  65001     sdb-rs1-s1-arbiter1 gamma-master-3
0                    ARBITER        ARBITER        true      true         N/A               local
2001:DB8:1:2::42  65001     sdb-rs1-s2-m1       delta-master-1
```

```
102        102       SECONDARY       SECONDARY       false      false         0.0                remote
==========================================================================================================
```

Note Follow a similar configuration for the Admin DB.

Configure SNMP alerts through IPv6 address

Follow these steps to configure SNMP alerts through IPv6 address. This includes configurations for SNMP v2 and v3 trappers.

Procedure

Step 1 Enable the SNMP trapper and set the v2c target with an IPv6 address. Configure source IP routes using internal and external IPv6 Virtual IP addresses (VIPs)

Example:

```
snmp-trapper enable true
snmp-trapper v2c-target 2001:DB8:1::116
 port      162
 community Re4D0nLy5TrinG
exit
snmp-trapper source-ip-routes internal-vip 2001:DB8:2::21
snmp-trapper source-ip-routes default-external-vip 2001:DB8:1::87
snmp-trapper source-ip-routes source-external-vips cee-alpha
 external-vip 2001:DB8:1::87
exit
```

Step 2 Enable the SNMP trapper and set the v3 target with an IPv6 address. Configure source IP routes using internal and external IPv6 VIPs.

Example:

```
snmp-trapper enable true
snmp-trapper v3-engine-id 80004f2222
snmp-trapper v3-target 2001:DB8:1::116
port      162
user-name root
auth      none
exit
snmp-trapper source-ip-routes internal-vip 2001:DB8:2::21
snmp-trapper source-ip-routes default-external-vip 2001:DB8:1::87
snmp-trapper source-ip-routes source-external-vips cee-alpha
external-vip 2001:DB8:1::87
exit
```

Dual-Stack deployment principles

The cnAAA supports both IPv4 and IPv6 concurrently. This dual-stack capability is fundamental to the configurations detailed in the Unified API and SNMP sections. Both IPv4 and IPv6 addresses are specified for network interfaces, VIPs, and service endpoints.

Key dual-stack configuration principles include:

- **Cluster mode**: The cluster is configured for `ipv6-mode dual-stack`.
- **Network interface configuration**: `netplan` configurations for VLANs and bonds include both IPv4 and IPv6 addresses and subnets.
- **Service endpoints**: Unified API external IP addresses, httproutes bind addresses, and RADIUS bind addresses are configured with both IPv4 and IPv6.
- **Virtual IP addresses**: Operations, Administration, and Maintenance (OAM) and Broadband Network Gateway (BNG) VIPs are defined with both IPv4 and IPv6 addresses.

IPv4 and IPv6 forwarding validation

Validation of both IPv4 and IPv6 forwarding is important for all relevant CPC services. This validation confirms that network traffic flows correctly over both protocols within the deployed environment..

How to validate

The successful execution of these key operations confirms correct network forwarding for both IPv4 and IPv6:

- **Cluster Deployment and Management**: Core cluster components deploy with dual-stack configurations. This ensures node-to-node communication.
- **SNMP Alerts and Communication**: SNMP alerts are received and processed through configured IPv6 trappers.
- **Geo-Redundancy GR with CDL**: Data replicates and synchronizes between different sites using IPv6 for CDL communication.
- **Geo-Redundancy GR with Mongo**: Data replicates and synchronizes between different sites using IPv6 for SPR or Admin DB communication.

Validate IPv6-enabled URLs

Verify access to Grafana, Policy Builder, Control Centre, and CPS Central through their respective IPv6-enabled URLs after the configurations are applied.

- Grafana: `https://grafana.2001-db8-1--87.sslip.io/`
- CPS Central:
  `https://pb.<namespace>-pcf-engine-app-production-rjio.2001-db8-1--87.sslip.io/central/#!/central`
- Policy Builder (PB):
  `https://pb.<namespace>-pcf-engine-app-production-rjio.2001-db8-1--87.sslip.io/pb`
- Control Center: `https://pcf.controlcenter.2001-420-2c7f-f6ad--87.sslip.io`

Cloud native CPS application on CNDP platform

Feature History

Feature Name	Release Information	Description
Cloud native CPS application on CNDP platform	2025.01.0	This feature introduces mechanisms to manage and protect the cnAAA Radius Engine from overload conditions by implementing message prioritization, message throttling, queue handling, and KPI monitoring, enhancing system stability and performance.

Overview

The cnAAA features a dedicated Radius Endpoint POD within its protocol layer, using a gRPC framework to handle Authentication, Accounting, and CoA requests. This setup ensures effective communication with the Policy Engine, enhancing message processing and session management. To maintain performance, cnAAA includes overload protection for handling message bursts. Kubernetes supports application deployment, offering zero downtime and automatic scaling to manage increased loads or recover from failures. Configuration involves software and code updates, with dynamic options for flexibility. This structure ensures robust and efficient network access management in cnAAA.

Configure the RADIUS Endpoint in cnAAA using Ops-Center

To configure the RADIUS Endpoint in cnAAA using Ops-Center, follow these steps:

Procedure

Step 1 Configure the RADIUS Device Group by defining shared secrets and loopback addresses for each BNG device.

```
radius device-group ASR9K
  default-shared-secret sh512
  default-coa-shared-secret aes256
  device dev1
    ip 10.1.2.12
    shared-secret aes128
    coa-shared-secret sh256
    loopback-addresses [12.3.1.2]
  exit
  device dev2
    ip 3.4.5.6
    shared-secret sh345
    coa-shared-secret aes111
    loopback-addresses [3.4.8.1]
  exit
```

Step 2 Configure the RADIUS Server Group by setting up primary and secondary server IPs, ports, server-type, and authentication protocols.

```
radius server-group grp1
  servers serv1
    primary 3.4.4.4
    secondary 5.5.5.3
    nas-ip 1.1.2.2
    accounting-port 8312
    authorization-port 1312
    auth-protocol PAP
    radius-password test123
    shared-secret sh233
    timeout-seconds 20
    test-message false
    test-userid test
    test-password test123
    thread-pool-size 10
    max-proxy-queue-size 3
    server-type   online (or) offline
  exit
```

Verification of Radius Endpoint configuration in cnAAA

Procedure

Verify the configuration by using the following command to ensure the running configuration matches the intended setup:

```
pcf# show running-config radius | nomore
```

The expected result is that the output should match the configured parameters for devices and server groups, confirming the setup is as intended.

Solution for Service Mismatch

Feature History

Feature Name	Release Information	Description
Solution for Service Mismatch	2025.02.0	This feature addresses issues caused by failures in CoA messages between cnAAA and BNG and vice-versa. It uses a backoff retry strategy to retry CoA messages at increasing intervals and handles error codes, targeting negative acknowledgment (NAK) for retries. The backoff Retry Strategy and Error Code Handling mechanisms prevent unauthorized access and ensure accurate management of subscriber services.

Overview

The Service Mismatch Handling feature in RADIUS Support for cnAAA ensures delivery of Change of Authorization (CoA) messages to the Broadband Network Gateway (BNG). CoA delivery failures can lead to unauthorized service access. This feature uses a retry mechanism with a backoff strategy, allowing the BNG time to recover and process CoA requests. The stack within the RADIUS Endpoint implements the backoff retry strategy.

CoA back-off retry in Ops-Center

To configure CoA backoff retry in Ops-Center, follow these steps:

Procedure

Step 1 Login to the Ops-Center.

Step 2 Enable and configure the CoA back off retry configurations with the following parameters:

- **backOffRetryCoA.maxRetransmission**

  It indicates the maximum number of times the cnAAA will retry the CoA. To disable, set N=0 (Default is 300; update as required).

  ```
  radius properties backOffRetryCoA.maxRetransmission value <N>
  ```

- **backOffRetryCoA.CoANackErrorCause**

Retry attempts are made when the cnAAA receives a CoA NAK from the BNG with error codes configured in this parameter. Any CoA NAK received by cnAAA other than these error codes are not retried. (Default value is 506,405,1001; update as required).

```
radius properties backOffRetryCoA.CoANackErrorCause value 506,405,1001
```

- **backOffRetryCoA.constantdelayInSeconds**

  If the interval between the back off retries is greater than the configured value, then this retry interval is used for further retries.(Default is 60; update as required).

```
radius properties backOffRetryCoA.constantdelayInSeconds value 60
```

Ops-Center configuration

To configure CoA parameters through the Ops-Center CLI, navigate to the RADIUS device-group configuration level. Define the timeout and retry values specifically for the ASR9K device group.

1. Enter the `config` mode for the specific RADIUS device group.
2. Define the CoA timeout, retries, and shared secret.

   Example:

```
[gamma/gamma-cncps] pcf(config)# radius device-group ASR9K
[gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-timeout-seconds 5
[gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-retries 3
[gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-port 3799
[gamma/gamma-cncps] pcf(config-device-group-ASR9K)# default-coa-shared-secret
<your_secret_key>
[gamma/gamma-cncps] pcf(config-device-group-ASR9K)# commit
```

Parameter descriptions:

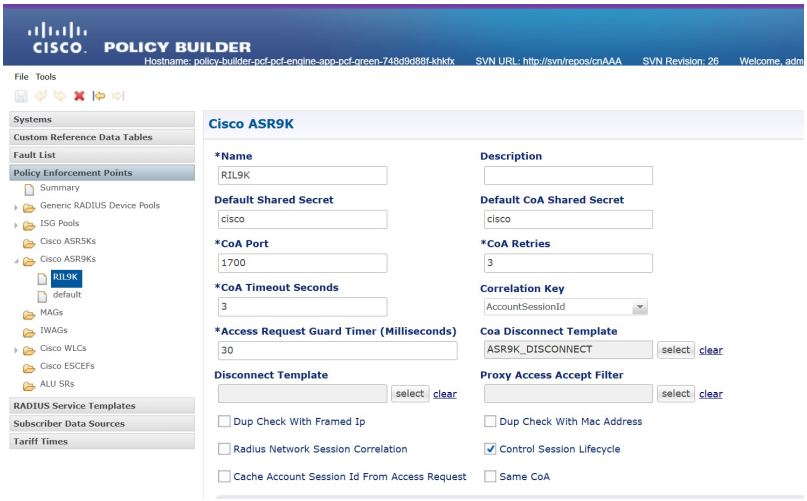
- **coa-timeout-seconds**: Specifies the duration, in seconds, that the CPC waits for a response from the ASR9K before timing out.
- **coa-retries**: Specifies how many times the CPC attempts to resend a CoA request if no response is received.
- **coa-port**: The port used for CoA messages.
- **default-coa-shared-secret**: This is the shared key used for authenticating CoA messages between the CPC and the ASR9K.

Policy Builder configuration

Synchronize the CoA parameters within the Policy Builder UI to ensure the policy engine correctly handles RADIUS client interactions.

Note Configure CoA timeout and CoA retry for each ASR9K device group in the system. This ensures consistency across the network.

Figure 8: Policy Builder RADIUS CoA settings



Procedure

Step 1 Log in to the Policy Builder.

Step 2 Navigate to **Reference Data** > **RADIUS** > **RADIUS Clients** (If you are using a different version, select the Device Group tab).

Step 3 Select the **ASR9K** device group from the list.

Step 4 Locate the **CoA Configuration** section within the device group settings.

Step 5 Update these fields:

- **CoA Timeout**: Enter the value in seconds.
- **CoA Retry**: Enter the number of retry attempts.

Step 6 **Save** and **Publish** the changes to the cluster.

CoA Request Retry Mechanism

The Back Off Retry involves multiple attempts with increasing delay times. Following table outlines the retry schedule:

Back Off Retry	Back Off Retry Delay in seconds
1st retry	1 + Timeout configured in Ops Center
2nd retry	3 + Timeout configured in Ops Center
3rd retry	7 + Timeout configured in Ops Center
4th retry	20 + Timeout configured in Ops Center
5th retry	55 + Timeout configured in Ops Center

6th retry	60 + Timeout configured in Ops Center

This process details the retry mechanism for sending Change of Authorization (CoA) requests from cnAAA to BNG when timeouts and NAK error codes occur. Assuming the CoA Retry Timeout is set to 3 seconds and the number of CoA Retries is configured to 3, the system will attempt retries based on these settings.

The following examples illustrate the timing of retries under different circumstances.

CoA Request Timed Out from BNG

This example shows the retry behavior when a CoA request times out due to no response from the BNG.

- **Initial CoA Request** (Retry attempted in RADIUS stack):
  - T0: CoA#1
  - T0+3: CoA#2 (retry#1)
  - T0+6: CoA#3 (retry#2)
  - T0+9: CoA#4 (retry#3)
  - CoA#4 will timeout at T0+12
- **1st Backoff Retry** (retry attempted after 1 second):
  - T0+12+1: CoA#5
  - T0+12+1+3: CoA#6
  - T0+12+1+3+3: CoA#7
  - T0+12+1+3+3+3: CoA#8
  - T0+12+1+3+3+3+3: CoA#8 will timeout
- **2nd Backoff Retry** (retry attempted after 3 seconds):
  - T0+25+3: CoA#9
  - T0+25+3+3: CoA#10
  - T0+25+3+3+3: CoA#11
  - T0+25+3+3+3+3: CoA#12
  - T0+25+3+3+3+3+3 (T0+40): CoA#12 will timeout

CoA Nack Error Code Configuration

When a CoA request is negatively acknowledged (NAK) by the BNG, retries are triggered with a backoff mechanism. The behavior is shown in the following example:

```
Configure NACK error codes in OpsCenter using: backOffRetryCoA.CoANackErrorCause = <Comma
Separated Nack Error Cause>
```

Example: `backOffRetryCoA.CoANackErrorCause = "506,405,1001"`

CoA NAK Error from BNG to cnAAA

- **Initial CoA Request**

- T0 – CoA#1
- T0+2: CoA NAK (error code: 506) (response received after 2 seconds)

- **1st Backoff Retry (retry attempted after 1 second)**
  - T0+2+1: CoA#2
  - T0+2+1+2 (T0+5): NAK (error code: 506) (after 2 seconds)

- **2nd Back off Retry (retry attempted after 3 seconds)**
  - T0+5+3: CoA#3
  - T0+5+3+2: NAK (response received after 2 seconds)

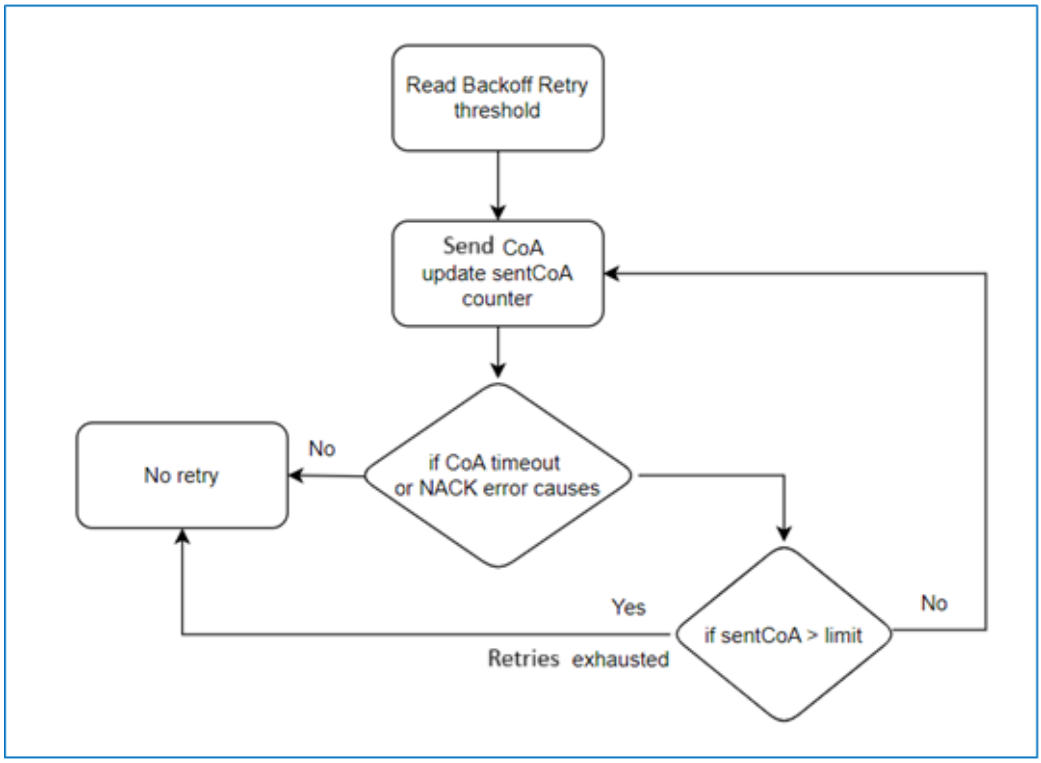
CoA BackOff Failure Report

- When the BackOff Retry exceeds the configured maxRetransmission limit, cnAAA updates the `coaBackOffFailureReport.csv` file located at `/data/consolidated-aaa-logging/` in the consolidated-aaa-logging pod with the following information:
  - Session Id
  - BNG IP Address
  - Subscriber Id
  - Service Details

```
2025-03-27 20:33:50,533  00012,11.11.90.72,0005.9A3C.7A00,
[subscriber:sd=A0F0500M050M000030MQ, subscriber:sa=A0F0500M100M000030MQ]
```

The flowchart explains the process of CoA retry mechanism, the interaction between the Engine, RADIUS Endpoint (EP), and BNG. The workflow includes handling retries, applying backoff logic, and determining success or failure based on responses.

Figure 9: CoA retry mechanism flowchart



1. The Engine initiates the CoA request workflow and reads the configured Backoff Retry Threshold to determine the maximum number of retry attempts, then sends the initial CoA request to the RADIUS Endpoint (EP).
2. The RADIUS EP forwards the CoA request to the BNG.
3. The BNG processes the CoA request and responds with one of the following:
   - ACK (acknowledgment): Indicates success.
   - NAK (negative acknowledgment): Indicates failure.
   - No response: Indicates timeout.
4. If the BNG responds with an ACK, the RADIUS EP forwards this to the Engine. The Engine considers the process successful, and the workflow ends.
5. If the BNG responds with a NAK or does not respond (timeout), the RADIUS EP forwards this response to the engine.
6. If a NAK or timeout is received, the engine updates the retry counter and, after a one second delay, either retries the CoA request (if within the retry threshold) or generates a failure report.
7. The workflow ends either upon success (ACK) or failure (retries exhausted).

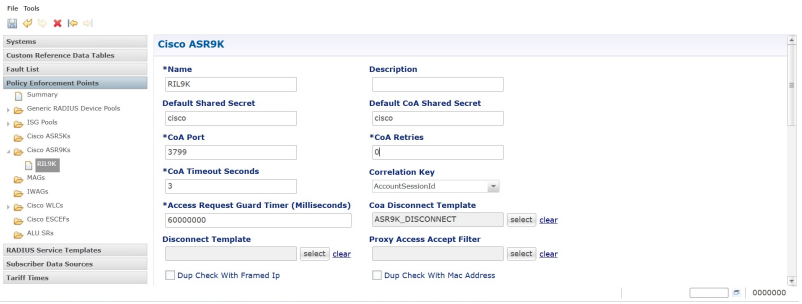
Configure gRPC timeout parameters

Follow these steps to configure the gRPC timeout properties:

Procedure

Step 1 Configure the CoA retry and timeout in CPC Ops-Center.

Figure 10: CoA retry and gRPC key




```
radius device-group ASR9K
 coa-port                    3799
 coa-retries                 0
 coa-timeout-seconds         3
```

Step 2 Calculate the timeout milliseconds value using this formula:

```
(CoA timeout seconds * (CoA retries + 1)) + 1|2 Seconds.
```

Step 3 Set the gRPC key timeout values using this commands:

```
radius properties grpc.timeoutMs.processing value 15000

engine <engine-name> properties grpc.request.asyncCoA.timeoutMillies value 15000

engine <engine-name> properties grpc.request.bundledCoA.timeoutMillies value 15000
```

Handling service mismatch between cnAAA and BNG

The service mismatch handling feature ensures the delivery of CoA messages to the BNG. This feature prevents unauthorized service access caused by delivery failures. The RADIUS endpoint stack uses a retry mechanism with a backoff strategy to allow the BNG to recover and process pending CoA requests.

CoA back-off retry in Ops-Center

To configure CoA backoff retry in Ops-Center, follow these steps:

Procedure

Step 1 Login to the Ops-Center.

Step 2 Enable and configure the CoA back off retry configurations with the following parameters:

- **backOffRetryCoA.maxRetransmission**

  It indicates the maximum number of times the cnAAA will retry the CoA. To disable, set N=0 (Default is 300; update as required).

  ```
  radius properties backOffRetryCoA.maxRetransmission value <N>
  ```

- **backOffRetryCoA.CoANackErrorCause**

  Retry attempts are made when the cnAAA receives a CoA NAK from the BNG with error codes configured in this parameter. Any CoA NAK received by cnAAA other than these error codes are not retried. (Default value is 506,405,1001; update as required).

  ```
  radius properties backOffRetryCoA.CoANackErrorCause value 506,405,1001
  ```

- **backOffRetryCoA.constantdelayInSeconds**

  If the interval between the back off retries is greater than the configured value, then this retry interval is used for further retries.(Default is 60; update as required).

  ```
  radius properties backOffRetryCoA.constantdelayInSeconds value 60
  ```

Ops-Center configuration

To configure CoA parameters through the Ops-Center CLI, navigate to the RADIUS device-group configuration level. Define the timeout and retry values specifically for the ASR9K device group.

1. Enter the `config` mode for the specific RADIUS device group.
2. Define the CoA timeout, retries, and shared secret.

   Example:

   ```
   [gamma/gamma-cncps] pcf(config)# radius device-group ASR9K
   [gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-timeout-seconds 5
   [gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-retries 3
   [gamma/gamma-cncps] pcf(config-device-group-ASR9K)# coa-port 3799
   [gamma/gamma-cncps] pcf(config-device-group-ASR9K)# default-coa-shared-secret
   <your_secret_key>
   [gamma/gamma-cncps] pcf(config-device-group-ASR9K)# commit
   ```

Parameter descriptions:

- **coa-timeout-seconds**: Specifies the duration, in seconds, that the CPC waits for a response from the ASR9K before timing out.
- **coa-retries**: Specifies how many times the CPC attempts to resend a CoA request if no response is received.
- **coa-port**: The port used for CoA messages.

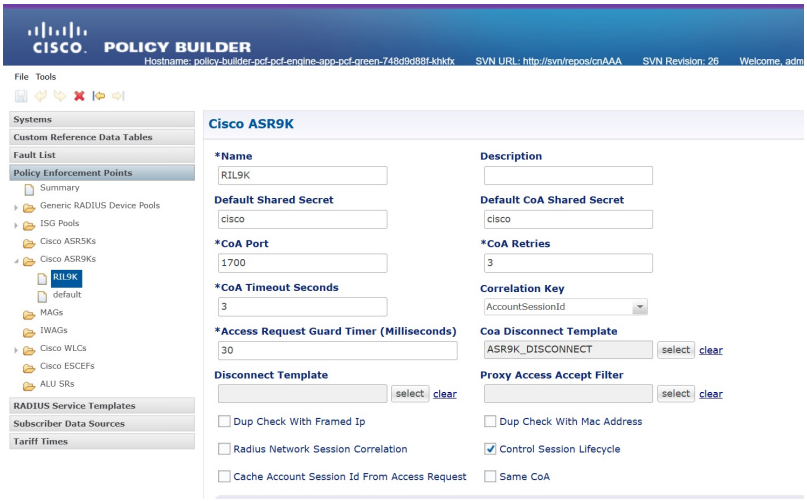
- **default-coa-shared-secret**: This is the shared key used for authenticating CoA messages between the CPC and the ASR9K.

Policy Builder configuration

Synchronize the CoA parameters within the Policy Builder UI to ensure the policy engine correctly handles RADIUS client interactions.

Note Configure CoA timeout and CoA retry for each ASR9K device group in the system. This ensures consistency across the network.

Figure 11: Policy Builder RADIUS CoA settings



Procedure

Step 1 Log in to the Policy Builder.

Step 2 Navigate to **Reference Data** > **RADIUS** > **RADIUS Clients** (If you are using a different version, select the Device Group tab).

Step 3 Select the **ASR9K** device group from the list.

Step 4 Locate the **CoA Configuration** section within the device group settings.

Step 5 Update these fields:

- **CoA Timeout**: Enter the value in seconds.
- **CoA Retry**: Enter the number of retry attempts.

Step 6 **Save** and **Publish** the changes to the cluster.

CoA Request Retry Mechanism

The Back Off Retry involves multiple attempts with increasing delay times. Following table outlines the retry schedule:

Back Off Retry	Back Off Retry Delay in seconds
1st retry	1 + Timeout configured in Ops Center
2nd retry	3 + Timeout configured in Ops Center
3rd retry	7 + Timeout configured in Ops Center
4th retry	20 + Timeout configured in Ops Center
5th retry	55 + Timeout configured in Ops Center
6th retry	60 + Timeout configured in Ops Center

This process details the retry mechanism for sending Change of Authorization (CoA) requests from cnAAA to BNG when timeouts and NAK error codes occur. Assuming the CoA Retry Timeout is set to 3 seconds and the number of CoA Retries is configured to 3, the system will attempt retries based on these settings.

The following examples illustrate the timing of retries under different circumstances.

CoA Request Timed Out from BNG

This example shows the retry behavior when a CoA request times out due to no response from the BNG.

- **Initial CoA Request** (Retry attempted in RADIUS stack):
  - T0: CoA#1
  - T0+3: CoA#2 (retry#1)
  - T0+6: CoA#3 (retry#2)
  - T0+9: CoA#4 (retry#3)
  - CoA#4 will timeout at T0+12
- **1st Backoff Retry** (retry attempted after 1 second):
  - T0+12+1: CoA#5
  - T0+12+1+3: CoA#6
  - T0+12+1+3+3: CoA#7
  - T0+12+1+3+3+3: CoA#8
  - T0+12+1+3+3+3+3: CoA#8 will timeout
- **2nd Backoff Retry** (retry attempted after 3 seconds):
  - T0+25+3: CoA#9
  - T0+25+3+3: CoA#10
  - T0+25+3+3+3: CoA#11
  - T0+25+3+3+3+3: CoA#12

- T0+25+3+3+3+3+3 (T0+40): CoA#12 will timeout

CoA Nack Error Code Configuration

When a CoA request is negatively acknowledged (NAK) by the BNG, retries are triggered with a backoff mechanism. The behavior is shown in the following example:

```
Configure NACK error codes in OpsCenter using: backOffRetryCoA.CoANackErrorCause = <Comma
Separated Nack Error Cause>
```

Example: `backOffRetryCoA.CoANackErrorCause = "506,405,1001"`

CoA NAK Error from BNG to cnAAA

- **Initial CoA Request**
  - T0 – CoA#1
  - T0+2: CoA NAK (error code: 506) (response received after 2 seconds)
- **1st Backoff Retry (retry attempted after 1 second)**
  - T0+2+1: CoA#2
  - T0+2+1+2 (T0+5): NAK (error code: 506) (after 2 seconds)
- **2nd Back off Retry (retry attempted after 3 seconds)**
  - T0+5+3: CoA#3
  - T0+5+3+2: NAK (response received after 2 seconds)

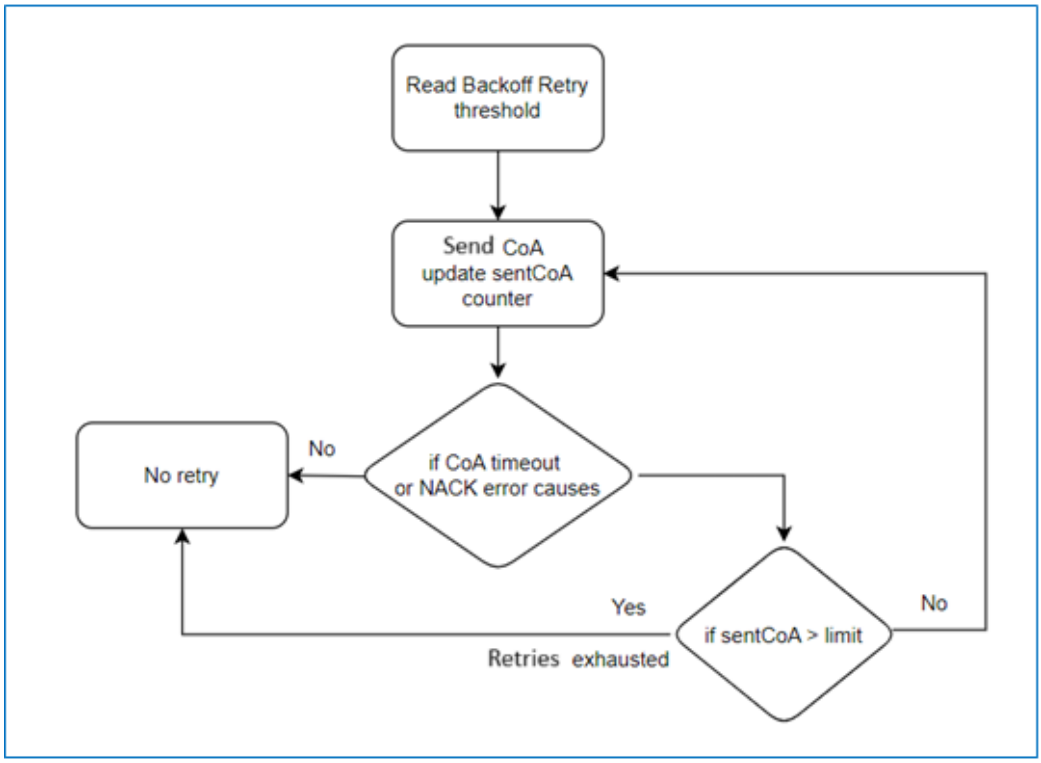
CoA BackOff Failure Report

- When the BackOff Retry exceeds the configured maxRetransmission limit, cnAAA updates the `coaBackOffFailureReport.csv` file located at `/data/consolidated-aaa-logging/` in the consolidated-aaa-logging pod with the following information:
  - Session Id
  - BNG IP Address
  - Subscriber Id
  - Service Details

```
2025-03-27 20:33:50,533  00012,11.11.90.72,0005.9A3C.7A00,
[subscriber:sd=A0F0500M050M000030MQ, subscriber:sa=A0F0500M100M000030MQ]
```

The flowchart explains the process of CoA retry mechanism, the interaction between the Engine, RADIUS Endpoint (EP), and BNG. The workflow includes handling retries, applying backoff logic, and determining success or failure based on responses.

Figure 12: CoA retry mechanism flowchart



1. The Engine initiates the CoA request workflow and reads the configured Backoff Retry Threshold to determine the maximum number of retry attempts, then sends the initial CoA request to the RADIUS Endpoint (EP).
2. The RADIUS EP forwards the CoA request to the BNG.
3. The BNG processes the CoA request and responds with one of the following:
   - ACK (acknowledgment): Indicates success.
   - NAK (negative acknowledgment): Indicates failure.
   - No response: Indicates timeout.
4. If the BNG responds with an ACK, the RADIUS EP forwards this to the Engine. The Engine considers the process successful, and the workflow ends.
5. If the BNG responds with a NAK or does not respond (timeout), the RADIUS EP forwards this response to the engine.
6. If a NAK or timeout is received, the engine updates the retry counter and, after a one second delay, either retries the CoA request (if within the retry threshold) or generates a failure report.
7. The workflow ends either upon success (ACK) or failure (retries exhausted).

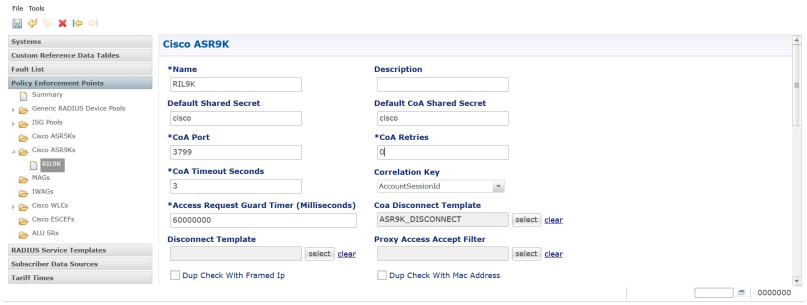
Configure gRPC timeout parameters

Follow these steps to configure the gRPC timeout properties:

Procedure

Step 1 Configure the CoA retry and timeout in CPC Ops-Center.

Figure 13: CoA retry and gRPC key




```
radius device-group ASR9K
 coa-port                   3799
 coa-retries                0
 coa-timeout-seconds        3
```

Step 2 Calculate the timeout milliseconds value using this formula:

```
(CoA timeout seconds * (CoA retries + 1)) + 1|2 Seconds.
```

Step 3 Set the gRPC key timeout values using this commands:

```
radius properties grpc.timeoutMs.processing value 15000

engine <engine-name> properties grpc.request.asyncCoA.timeoutMillies value 15000

engine <engine-name> properties grpc.request.bundledCoA.timeoutMillies value 15000
```

Consolidation of CoA processing in a RADIUS endpoint for a given BNG

Feature History

Feature Name	Release Information	Description
Consolidation of CoA Processing in a RADIUS EP for a Given BNG	2025.02.0	This feature enhances CoA request handling for BNGs by using a NAS IP-based hashing mechanism to direct requests to a consistent RADIUS endpoint, preventing overloads.

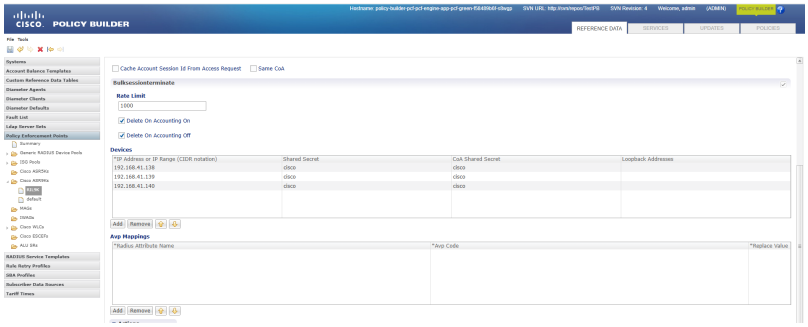
Overview

This feature optimizes CoA request handling for BNGs by directing requests from an engine pod to a consistent RADIUS endpoint using a hash derived from the NAS IP address of the BNG. The engine generates a hash number from the NAS IP Address, ensuring consistent endpoint selection for each BNG. This prevents overloads by avoiding inconsistent throttling that can occur with random distribution of CoA requests across endpoints.

BNG IPs is configured through Policy Builder and Ops Center.

BNG IP configuration in Policy Builder

Set up BNG IP addresses in the Policy Enforcement Point (PEP) of Policy Builder to enable RADIUS packet generation, allowing the Engine to recognize the configured BNG IPs.



Follow these steps to configure BNG IP addresses:

Procedure

Step 1 Login to **Policy Builder** and navigate to the **Policy Enforcement Point** > **RIL9K**.

Step 2 In the Devices section, click **Add** to add the BNG IP address.

Step 3 Click **File** on the top left corner of the Policy Builder GUI and select **Publish to Runtime Environment.**

BNG IP Configuration in Ops Center

BNG Configuration

Configure the BNG-IP and device details using the following setup:

Device Group Setup:

```
radius device-group ASR9K
device dev1
ip  192.68.41.138
shared-secret cisco
coa-shared-secret cisco
loopback-addresses [ 12.3.1.2 ]
exit
device dev2
ip  192.68.41.139
shared-secret cisco
coa-shared-secret cisco
loopback-addresses [ 12.3.1.2 ]
exit
device dev3
ip 192.68.41.140
shared-secret cisco
coa-shared-secret cisco
loopback-addresses [ 12.3.1.2 ]
exit
```



Note Adding a BNG IP address in Ops-Center causes the RADIUS Pod to restart. Complete this activity during the maintenance window.

Enable the CoA Processing Per BNG

Enable CoA processing per BNG in the Ops Center by setting the property value to true. The default configuration sets this property to false.

```
engine cpc-green
properties coaThrottlingPerBng
value true
exit
```

Configure CoA Throttling

Enable the following Ops Center configuration for the throttling feature.

```
radius advance-tuning throttling-limit 15

radius advance-tuning coa-throttlling-for-asr9k-pep true
```

Add/Remove the Radius Replicas

Use the following Ops Center configuration to modify the radius replicas.

```
radius replicas 2
```

CoA Request Scenarios in cnAAA

CoA Request Throttling

When the cnAAA engine receives a plan change request from the OCS, it initiates a process towards the RADIUS endpoint (radius-ep) to send a Change of Authorization (CoA) to the designated BNG. The engine generates a hash number from the NAS IP Address, ranging from 0 to N, representing the number of radius endpoints. This hash maps to the corresponding radius endpoint, allowing the engine to send a CoA Request for further processing to the configured BNG.

Example: If the cnAAA engine processes 20 CoA requests with a throttling value of 15, 15 requests will be successful while 5 are dropped due to throttling. The engine will retry the throttled requests and consistently route them to the radius-ep.

Endpoint Pod Changes

While adding or removing RADIUS endpoint pods, re-hashing is performed. As a result, CoA requests are always redirected to the correct RADIUS endpoint based on the hash value of the configured BNGs. The hashing mechanism minimizes key redistribution when the number of RADIUS Endpoint pods changes, focusing on maintaining consistent routing rather than equal distribution of keys.

Combined multi-CoA support

The Policy Builder GUI supports configuration of the entire cnAAA virtual machine cluster, services, and advanced policy rules. Combined CoA support enables validation of activating and deactivating up to five services with a single CoA.

Configure multiple CoA in a radius message

These steps describe the procedure to combine multiple CoA in a radius message:

Procedure

Step 1 Open the Policy Builder GUI using these steps:

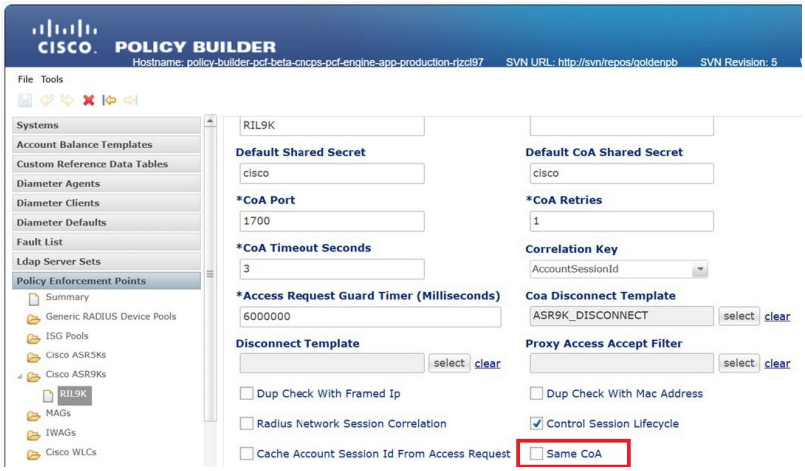
a) Execute this command: `kubectl get ing -n <cpc namespace> | grep pb`

```
policy-builder-httproutes-pcf-beta-cncps-pcf-engine-app-production-rjio nginx
pb.<namespace>-pcf-engine-app-production-rjio.<ip>.nip.io <ip>
```

b) Build the PB URL: *`https://pb.<namespace>-pcf-engine-app-production-rjio.<ip>.nip.io/pb`*

Step 2 In the Policy builder, choose the **Policy Enforcement Points** > **Cisco ASR9K**.

Step 3 Select the same **CoA checkbox** to enable the Change of Authorization (CoA) so that multiple services are handled in a single CoA request.



Note Multi-CoA will impact the overall TPS on the BNG. Turn on the feature only after you consult the BNG team.

CoA Request and Response Metrics

These metrics provide an overview into the processing and outcomes of CoA requests and responses:

Throttled CoA Requests:

Description: Indicates the total number of CoA requests that were throttled due to exceeding set limits.

```
radius_coa_requests_throttled_total{message_type="CoARequestThrottle",
nas_ip_address="192.168.41.138",endpoint_address="192.102.6.99",} 5.0
```

Total CoA Requests:

Description: Represents the total CoA requests processed.

```
radius_coa_requests_total{message_type="CoARequest",
nas_ip_address="192.168.41.138",endpoint_address="192.102.6.99",retry_type="NA",
result="SUCCESS",} 20.0
```

CoA Acknowledgment Responses:

Description: Captures acknowledgment responses for CoA requests. The metric shows a successful CoA ACK response from NAS IP address to endpoint, with no NAK error cause specified.

```
radius_coa_responses_total{message_type="CoAACKResponse",
nas_ip_address="192.168.41.138",client_ip_address="192.168.101.70",
endpoint_address="192.102.6.99",nak_error_cause="",}
```

User Plane IP (vBNG) support

Feature Name	Release Information	Description
User Plane IP (vBNG) support	2026.02.0	This feature enables the system to carry the User Plane (UP) IP address within RADIUS accounting messages, decoupling the Control Plane from the physical User Plane infrastructure to enhance scalability and resolve unfair throttling. The RADIUS endpoint parses the new IP Attribute-Value Pair (AVP) from accounting requests and propagates it to the backend policy engine, which updates the session record in the CDL.

This feature carries the User Plane (UP) IP address in RADIUS accounting messages. This enhances network scalability, abstraction, and management in the Control Plane (CP). The system decouples the Control Plane from the underlying physical UP BNG infrastructure by assigning a stable, virtual IP address to each UP BNG. The Control Plane BNG uses this User Plane IP as a consistent endpoint for communication with the cnAAA server.

The system supports these two Vendor-Specific Attributes (VSAs):

- user-plane-ipv4-address: Carries an IPv4 User Plane IP.
- user-plane-ipv6-address: Carries an IPv6 User Plane IP.

Note Only one attribute is present per session message. If both are present, the system defaults to the IPv4 attribute.

User Plane IP processing

The system processes, stores, and updates the UP IP address from RADIUS requests through these stages:

1. **Request transmission**: The BNG sends a RADIUS request containing the User Plane IP AVP.
2. **Parsing**: The RADIUS EP parses and extracts the UP IP address.
3. **Propagation**: The RADIUS EP propagates the IP address to the backend policy engine.
4. **Persistence**: The policy engine updates the corresponding session record in the CDL.
5. **Response**: The RADIUS EP sends an accounting response back to the BNG.

Note The system generates `DEBUG` level logs upon successful parsing and storage of the UP IP address. It generates `ERROR` level logs if the AVP is present but fails to parse, or if the CDL update fails.

CoA throttling based on User Plane IP

Feature Name	Release Information	Description
CoA throttling based on User Plane IP	2026.02.0	This feature provides granular traffic management by using the unique User Plane IP address as the primary key for throttling decisions. This addresses the unfair throttling by isolating and throttling only the specific User Plane BNG that exceeds its rate limits.

The legacy CoA throttling mechanism identifies all traffic by the Control Plane (CP) BNG address, which prevents the system from distinguishing between individual User Plane (UP) BNGs. As a result, a traffic surge or misconfiguration in one UP BNG causes the system to throttle CoA requests for all other healthy UP BNGs sharing the same Control Plane.

The CoA throttling based on User Plane IP feature uses the unique User Plane IP address as the primary key for throttling decisions. This allows the system to isolate and throttle only the specific UP BNG that exceeds its rate limits, preventing collateral impact on other BNGs managed by the same Control Plane.

Prerequisites

Before you enable CoA throttling based on User Plane IP, ensure that you meet these requirements:

- Ensure the cnAAA platform is upgraded to the release supporting CoA throttling based on User Plane IP.
- Ensure all RADIUS engine and RADIUS endpoint pods are in a stable, running state.
- Enable the CoA throttling feature and set the required properties in Ops-Center.

```
radius advance-tuning coa-throttlling-for-asr9k-pep true <To enable the coa throttlling>
radius properties com.broadhop.minimum.throttling.limit value 10 <Default value is 10>
engine <engine-name>  properties coaThrottlingPerBng value true
```

Troubleshoot

If the system does not throttle based on the expected User Plane IP, perform these checks:

1. Ensure the upstream system correctly populates the `user-plane-ipv4-address` or `user-plane-ipv6-address` RADIUS attributes.
2. Check whether the User Plane IP address is stored in the session or not either using Control center or getSubscriber API.
3. Check the `consolidated-app.log`and `consolidated-engine.log` for errors related to the `CoAThrottlingTimer` or gRPC message parsing.

Note If the `user-plane-ip-address` attribute is absent in a request, the system automatically reverts to the legacy `NAS-IP-Address` throttling logic. This fallback ensures service continuity even if upstream systems are not updated to support the new attributes.

Virtual NAS-IP label for CoA KPIs

Feature description

Table 15:

Feature Name	Release Information	Description
Virtual NAS-IP label for CoA	2026.03.0	The virtual NAS IP enhancement introduces a logical label for RADIUS CoA to enable traffic management in disaggregated BNG architectures. By extracting the User Plane IP address from RADIUS attributes, the cnAAA adds the label `user_plane_ip_address`. This allows for targeted throttling and improved operational visibility without impacting healthy network nodes.

This enhancement integrates the `user-plane-ip-address` label into existing Change of Authorization (CoA) KPI metrics. By decoupling the Control Plane (CP) from the physical User Plane (UP) infrastructure, the cnAAA adds the `user_plane_ip_address` label. This improves monitoring, troubleshooting, and traffic throttling in disaggregated BNG architectures by enabling operational visibility and targeted traffic management.

Monitor KPI updates

The cnAAA updates these existing metrics to include the `user_plane_ip_address` label. If the User Plane IP is present in a request, the system populates the label with the IP address. Otherwise, the system sets the label to an empty string or `none`.

Table 16: Updated KPI metrics

Metrics name	Description
`RADIUS_COA_RESPONSES_SECONDS`	Total RADIUS CoA response time in seconds.
`RADIUS_COA_REQUESTS`	Total Radius CoA requests.
`RADIUS_COA_RESPONSES`	Total Radius CoA responses.

Metrics name	Description
RADIUS_COA_REQUEST_TIMEOUT	Total RADIUS CoA request timeout.
RADIUS_COA_REQUEST_TIMEOUT_SECONDS	Total RADIUS CoA request timeouts in seconds.
RADIUS_COA_REQUEST_ERROR	Total RADIUS CoA errors.
RADIUS_COA_THROTTLE_REQUESTS	Total number of CoA requests throttled.

Verify traffic processing

Use this procedure to validate that the system receives and processes the User Plane IP.

Procedure

Step 1 Use the **Control Center** or the `getSubscriber` API to confirm that the `user-plane-ip-address` is correctly populated in the subscriber session record.

Step 2 Execute this command on the master node to verify that the metrics are exported with the new label:

```
kubectl exec -it <engine-pod> -n <namespace> -- curl -G http://127.0.0.1:8080/metrics | grep
user_plane_ip_address
```

Step 3 Update existing panels to filter by the `user_plane_ip_address` label to view traffic distribution per logical User Plane entity.

Troubleshoot

Use this information to identify and resolve common issues with virtual NAS-IP label support.

Table 17: Troubleshooting vBNG support

Issue	Corrective action
Throttling not occurring as expected	Verify that the upstream BNG correctly sends the `Cisco-AVPair` attribute. If the attribute is absent, the system reverts to the legacy `NAS-IP-Address` throttling logic.
Missing labels in KPIs	Ensure the cnAAA application version is updated. Check `consolidated-app.log` and `consolidated-engine.log` for errors related to `CoAThrottlingTimer` or gRPC message parsing.
Attribute parsing errors	If the AVP is present but fails to parse, the cnAAA RADIUS endpoint generates an `ERROR` level log. Inspect the logs for malformed AVP values.

CoA packet authenticator generation

Feature description

Table 18: Feature description

Feature title	Release information	Description
RADIUS packet message authenticator	2026.03.0	This feature updates the RADIUS packet assembly logic to ensure full RFC compliance. It enforces a correct cryptographic sequence by finalizing all mutable fields before computing the `Request-Authenticator`. Additionally, it introduces active validation and logging to monitor authenticator integrity and simplify troubleshooting.

The cnAAA RADIUS Endpoint has been updated to fix the CoA packet assembly logic. This change ensures that the Request-Authenticator is computed as the last encoded field. It resolves cryptographic mismatches that previously caused legitimate CoA-Requests to be rejected by downstream BNG or `FreeRADIUS` devices. This enhancement finalizes all mutable fields—including the `Message-Authenticator` before computing the `Request-Authenticator`. As a result, it guarantees compliance with RFC 5176 and RFC 2865/2866/2869, ensures cryptographic integrity and interoperability, and prevents unnecessary packet rejections due to authenticator validation failures.

To ensure RFC 5176 compliance, the `Request-Authenticator` must be computed only after all mutable fields (including the `Message-Authenticator`) are finalized:

Target compliant flow:

1. Build CoA packet skeleton.
2. Set `Request-Authenticator` placeholder = 0x00…00.
3. Set `Message-Authenticator` placeholder = 0x00…00.
4. Compute `Message-Authenticator` (HMAC-MD5 over packet with MA zeroed and RA placeholder).
5. Insert `Message-Authenticator`.
6. Compute `Request-Authenticator` (MD5 over final packet + shared secret per implementation rules).
7. Insert `Request-Authenticator`.
8. Send packet.

Configure the CoA packet authenticators

Before you begin

Verify the RADIUS endpoint, ensure that you meet these requirements:

- Verify that all cnAAA RADIUS endpoint pods are running and in a healthy state.
- Confirm that the `shared-secret` and `coa-shared-secret` are correctly provisioned across the cnAAA and the target NAS or BNG devices.

When deployed, the cnAAA RADIUS endpoint automatically applies the corrected packet assembly logic. You must explicitly enable the validation logic and logging using the Ops-Center CLI.

Procedure

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

Step 2 Enable the RADIUS authentication property.

```
radius properties enable.radius.auth
value true
exit
```

Step 3 Enable the `Message-Authenticator` for all RADIUS message types:

```
radius message-authenticator access-accept true
radius message-authenticator access-request true
radius message-authenticator accounting-request true
radius message-authenticator accounting-response true
radius message-authenticator coa-request true
radius message-authenticator coa-response true
radius message-authenticator access-reject true
```

Step 4 Configure the endpoint logger to capture validation events.

```
debug logging logger com.cisco.radius.endpoint.impl.RadiusEndpointMessageListener level
      warn
```

Step 5 `commit` the changes.

Step 6 Verify the status of the RADIUS Endpoint pods.

```
kubectl get pods -n <namespace> | grep radius-ep
```

Confirm that all RADIUS Endpoint pods are in the `Running` state and that the `READY` column indicates all containers are active (e.g., `1/1`).

Step 7 Ensure successful acknowledgments (ACKs) from the BNG:

```
radius_coa_responses_total{message_type="CoAACKResponse", result="SUCCESS"}
```

Step 8 Check the `consolidated-app.log` for any authenticator-validation warnings:

```
kubectl logs -n <namespace> consolidated-logging-0 | grep -i "RadiusEndpointMessageListener"
```

Interpret validation logs:

- Missing attribute: If a packet lacks the `Message-Authenticator`, the system logs:

```
WARN c.c.r.e.i.RadiusEndpointMessageListener - Message Authenticator validation is
          enabled but received accounting request does not have
        message-authenticator...
```

- Failed validation (Invalid HMAC): If the HMAC-MD5 check fails, the packet is dropped:

```
WARN c.c.r.e.i.RadiusEndpointMessageListener - Dropping the AccountingRequest as
        MessageAuthenticator check is failing...
```

Note Setting the level to WARN lets you see validation failures and bypass messages.

Monitoring

The cnAAA RADIUS Endpoint provides specific KPIs to track invalid **Message-Authenticator** events across different RADIUS message types:

KPI Metric	Labels
`radius_invalid_message_authenticator_coa_request_total`	request_type, nas_ip_address, client_ip_address, endpoint_ip_address, result
`radius_invalid_message_authenticator_coa_response_total`	request_type, nas_ip_address, client_ip_address, endpoint_ip_address, result
`radius_invalid_message_authenticator_accounting_request_total`	accounting_type, status_type, clientIp, endPointIp, result
`radius_invalid_message_authenticator_access_request_total`	message_type, nas_ip_address, client_ip_address, endpoint_address, result

Troubleshoot

Use this table to identify and resolve common issues related to CoA request rejections.

Table 19: Troubleshooting CoA-Request rejections

Symptom	Possible cause	Corrective action
CoA-Requests continue to be rejected by BNG.	Shared secret mismatch.	Verify the `shared-secret` and `coa-shared-secret` in the `radius device-group` configuration in the Ops-Center.
Authenticator calculation failures in logs.	Packet assembly issue.	Review `consolidated-engine.log` for Attribute-Value Pair (AVP) parsing errors. Ensure the system does not modify AVPs after the RA computation.
High rate of CoA timeouts.	BNG connectivity or throttling.	Check `radius_coa_requests_throttled_total` metrics to determine if the BNG is throttling requests.

Circle code labels for RADIUS KPIs

Circle code labels for RADIUS KPIs

The RADIUS endpoint module supports geographic traffic identification by adding a `circle_code` label to all RADIUS-related KPIs. This label is dynamically extracted from the `CALLED_STATION_ID` Attribute-Value Pair (AVP) in inbound RADIUS messages.

How Circle Code extraction works

The RADIUS endpoint performs these action the `circle_code` value from the `CALLED_STATION_ID` string:

- Extracts the characters at index 2 and 3 (the 3rd and 4th characters) from the `CALLED_STATION_ID` value.
- Converts the extracted value to uppercase to ensure consistency across metrics.
- Sets the `circle_code` label to `UNKNOWN` if the `CALLED_STATION_ID` AVP is absent, null, or contains fewer than four characters.

RADIUS KPI Metrics with Circle Code

These metrics include the `circle_code` label for geographic traffic identification:

Category	Supported Metrics
General requests	`inbound_request_total`, `outbound_request_total`, `radius_requests_total`, `radius_responses_total`
Errors and timeouts	`radius_request_error_total`, `radius_request_timeout_total`, `radius_requests_packet_error_total`, `radius_late_responses_total`, `radius_discard_requests_total`
CoA metrics	`radius_coa_requests_total`, `radius_coa_responses_total`, `radius_coa_request_error_total`, `radius_coa_request_timeout_total`, `radius_coa_requests_throttled_total`
Accounting metrics	`radius_accounting_request_total`, `radius_accounting_response_total`, `radius_accounting_error_total`, `radius_accounting_response_timeout_total`
Processing	`radius_ep_request_processing_total`

Monitoring tool configuration and verification

Use these procedures to integrate the `circle_code` label into your monitoring tools and verify the metric population.

Update Grafana dashboards

To visualize traffic per geographic region, update your existing panels to include the `circle_code` label in the query aggregation.

1. In the Grafana panel editor, modify the PromQL query to group by the new label.

- Before:

```
sum(rate(radius_accounting_request_total[1m])) by (pod)
```

- After:

```
sum(rate(radius_accounting_request_total[1m])) by (pod,
        circle_code)
```

2. Add a dashboard variable named `circle_code` with the query `label_values(radius_requests_total, circle_code)` to allow users to filter traffic for specific regions dynamically.

Update Prometheus alert rules

Review your existing alert rules. If an alert is global, aggregate away the new label to prevent the alert from splitting into multiple, smaller alert instances.

- Use the `sum without` operator if an alert should trigger regardless of the circle.

```
sum without (circle_code) (rate(radius_request_timeout_total[5m])) >
          100
```

- Define a rule with a label matcher to create alerts for specific regions (for example, detecting a BNG failover in a specific circle).

```
rate(radius_request_timeout_total{circle_code="KA"}[5m]) >
      50
```

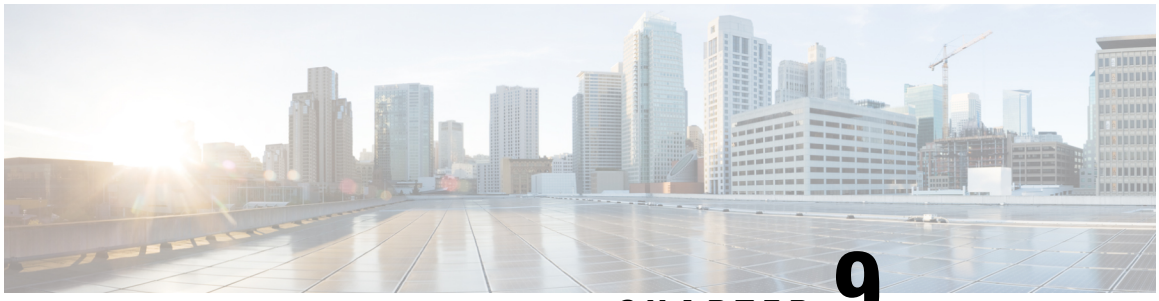
Verify and troubleshoot metrics

Use this procedure to validate the metric population and monitor system performance.

1. Execute these command on the master node to verify that the label is present in the metrics stream:

```
kubectl exec -it <radius pod name> -n <namespace> -- curl -G http://127.0.0.1:9099/metrics
 | grep circle_code
```

2. If you see many `circle_code="UNKNOWN"` labels, check the BNG device configuration. Make sure that the BNG devices provide the `CALLED_STATION_ID` AVP and that this AVP is at least four characters long.
3. Monitor the `prometheus_tsdb_head_series` metric to ensure that the total number of series remains within the cluster memory capacity.



CHAPTER 9

Persistent Storage for Policy Configuration

- Overview, on page 193
- How it works, on page 193
- Configure Persistent Storage, on page 194
- Configure the Restore Capability, on page 195

Overview

Persistent storage is a storage solution that retains the data after the power and network resources are disconnected.

The cnAAA provides various storage technologies for managing the configuration data. The cnAAA has pre-defined storage such as the OpenStack Cinder volume used for storing the CRD data. cnAAA 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, cnAAA uses the local storage class along with the default storage layer as the persistent storage.

For generic information on the Persistent Volume concepts, see the Kubernetes documentation.

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 protects the configuration data and maintains its integrity when the SVN pod restarts.

How it works

The cnAAA implements the Kubernetes Persistent Volume (PV) framework, which lets the administrators allocate persistent storage for a cluster. Regardless of the storage tier, you can use the Persistent Volume Claims (PVCs) to request PV resources. You must enable persistent volume claim and assign storage that represents local storage. The data residing on the local storage is intact in situations where the associated node or pod restarts.

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 the following configMaps:

- **Monitor-svn-configmap-cnAAA**: 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

This section describes how to configure persistent storage.

Configure the persistent storage in cnAAA involves these steps:

- Enable Support for Persistent Storage
- Assign Persistent Storage

Enable Support for Persistent Storage

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

1. To enable persistent volume claim, use the following configuration:

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

NOTES:

- **config**—Enters the configuration terminal.
- **k8s**—Enters the Kubernetes configuration mode.
- **use-volume-claims [true | false]**—Configures using the volume claims during the NF deployment. When set to true, the default storage class such as OpenStack Cinder is enabled. If the **use-volume-claims** is set to false, then the data gets stored in the memory that is susceptible to lose on a pod restart.

Assign Persistent Storage

Assign a storage volume as the ersistent storage using the CLI commands.

Before configuring the persistent storage, ensure that **use-volume-claims** is enabled.

To assign persistent storage, use the following configuration:

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

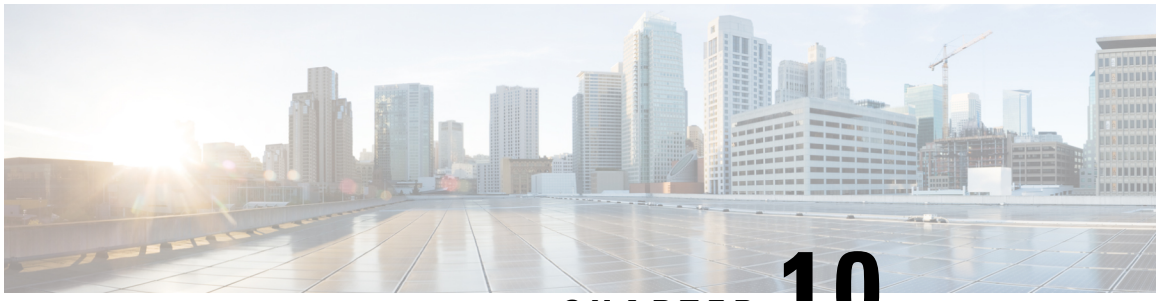
Configure the Restore Capability

This section describes how to configure the restore capability. To configure the restore capability that ensures the persistency of policy configuration file, use the following configuration in the Policy Ops Center console:

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

NOTES:

- **engine** *engine_name*—Specify the engine for which the restore capability must be configured
- **cnAAA policy-configuration-restore [true | false]**—Configures the capability that is responsible for restoring the configMap. The default value for this parameter is true.



CHAPTER 10

Managing Custom Reference Data

- Feature Description, on page 197
- Configuration support for importing CRD, on page 197
- CRD refresh interval configuration and performance improvements in CRD functionality, on page 214

Feature Description

The Custom Reference Data (CRD) is the reference data specific to a service provider, such as their networks or cell sites' names and characteristics. This data is required to operate the policy engine but not used for evaluating the policies. The CRD is represented in the table format. The service providers have the flexibility to create custom data tables and manage them as per their requirements.

Note Make sure to start all the policy servers after a CRD table schema is modified (for example, column added/removed).

CRD supports the pagination component, which controls the data displayed according to the number of rows configured for each page. You can change the number of rows to be displayed per page. Once you set the value for rows per page, the same value is used across the Central unless you change it. Also, you can navigate to other pages using the arrows.

Configuration support for importing CRD

This section describes the procedure to import CRD when the CRD schema is modified.

Importing of CRD involves the following 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

Backup the existing SVN repository

This section describes how to import the SVN repository when the CRD schema is modified.

To take a backup of the existing SVN repository and store it on another environment, use the following configuration:

1. Log in to the cnAAA Central GUI.
2. On the **Converged Policy & Charging Central** page, navigate to **Policy Builder** and click the **Import/Export** link.

   The **Import/Export** form opens.
3. In the **Export** tab, select the **All data** option to configure the export type.

   The following table describes the export/import options:

   Table 20: Export and Import Options

Parameters	Description
All data	Exports service configuration with environment data, which acts as a complete backup of both service configurations and environmental data.
Exclude Environment	Exports without environment data, which allows exporting configuration from a lab and into another environment without destroying the new system's environment-specific data.
Only Environment	Exports only environment data, which provides a way to back up the system-specific environmental information.
Export URL	The URL can be accessed from the Policy Builder or viewed directly in Subversion.
Export File Prefix	Provide a name (prefix) for the export file. **Note** The exported filename automatically includes the date and time when the export was performed.

4. If you want to export the file in the compressed format, select the **Use 'zip' file extension** check box.
5. Click **Export**.
6. Navigate to the file and save it to your local machine. The file must include the cluster name and date.

Backup the existing CRD

This section describes how to import an existing CRD when the CRD schema is modified.

To take a backup of the configured CRD and store it to another environment, use the following configuration:

1. Log in to the cnAAA Central GUI.
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.
3. Under **Export Custom Reference Data**, the following options are displayed:

   Table 21: Export Custom Reference Data Options

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

4. Click **Export**.

Remove the existing CRD from MongoDB

This section describes how to remove the existing CRD tables that have schema change from MongoDB.

To remove a configured CRD schema change, use the following configuration:

1. Log in to the admin-db pod that has the CRD (cust_ref_data) database.
2. Access the cust_ref_data using the following command:

   ```
   use cust_ref_data
   ```
3. Delete the data from one or more existing CRD tables using the following command:

   ```
   db.table_name.remove({})
   ```
4. Exit the admin-db pod.

Mongo Replica Sets configuration

Mongo replica set configurations support both IPv4 and IPv6 addresses for replica member hosts. This ensures that the initialization, replication, and initial sync functionalities of the Mongo replica set operate without any impact, regardless of whether IPv4 or IPv6 addresses are used. This dual support allows for flexible network configurations and aids in transitioning to IPv6 while maintaining compatibility with existing IPv4 infrastructure.

Each replica subscriber must be configured with either IPv4 or IPv6 addresses to ensure optimal functionality. It is not recommended to mix both IP versions within a single subscriber configuration.

Ops-Center configuration for enabling IPv6 in MongoDB Replica Sets

To enable IPv6 in MongoDB replica sets within CPC, following configurations in OpsCenter are required:

Database and Replica Set Initialization

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

Replica Set Labels

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

Member Configurations:
Member: sprdb-rs1-arbiter

Host: 2001:ddff::8
Arbiter: true
Site: remote
Member: sprdb-rs1-s1-m1

Host: 2001:ddff::9
Arbiter: false
Priority: 166
Site: remote
Member: sprdb-rs1-s1-m2

Host: 2001:ddff::10
Arbiter: false
Priority: 155
Site: remote
Member: sprdb-rs1-s2-m1

Host: 2001:ddff::11
Arbiter: false
Priority: 66
Site: local
Member: sprdb-rs1-s2-m2

Host: 2001:ddff::12
Arbiter: false
Priority: 55
Site: local
```

Import and publish the new CRD schema

This section describes how to import and publish the new CRD schema.

To import and publish the CRD schema, use the following configuration:

1. Log in to the cnAAA Central GUI.
2. On the **Converged Policy & Charging Central** page, navigate to **Policy Builder** and click the **Import/Export** link.

   The **Import/Export** form opens.
3. In the **Import** tab, browse to the file that you want to import.
4. In the **Import URL** field, enter the URL where the file must be imported. We recommend importing a new URL and verify it using the Policy Builder.
5. In the **Commit Message** field, enter the appropriate information.

6. To enforce import in situations where the checksums don't match, select the **Force import even if checksums don't match** check box.
7. Click **Import**.

Importing the New CRD

To import the new CRD, use the following configuration:

1. Access the Policy Builder URL and add a new repository.
   a. In the **Choose Policy Builder data reposiorty...** window, select **<Add New Repository>** from the drop-down.

      The **Repository** dialog box appears.

      The following parameters can be configured under **Repository**:

      Configure the parameters according to the network requirements.

      Table 22: Repository Parameters

Parameter	Description
Name	This is a mandatory field. Ensure that you specify a unique value to identify your repository's site. **Note** We recommend the following format for naming the repositories: customername_project_date, where underscores are used to separate customer name, project, and date. Date can be entered in the MMDDYYYY format.
Username and Password	Enter a username that is configured to view the Policy Builder data. The password can be saved for faster access, but it is less secure. A password, used with the Username, permits, or denies access to make changes to the repository.
Save Password	Select this check box to save the password on the local hard drive. This password is encrypted and saved as a cookie on the server.
Url	You can have several branches in the version control software to save different versions of configuration data. Create a branch in the version control software before assigning the URL in this screen. Enter the URL of the branch of the version control software server that is used to check in this version of the data.

Parameter	Description
Local Directory	Do not modify the value in this field. This is the location on the hard drive where the Policy Builder configuration objects are stored in the version control. When you click either Publish or Save to Repository, the data is saved from this directory to the version control application specified in the Url text field. The field supports the following characters: • Uppercase: A to Z • Lowercase: a to z • Digits: 1–9 • Nonalphanumeric: / **Note** The user must use only the supported characters.
Validate on Close	Select this check box to see if the values for Username, Password, or the URL are legitimate and unique. If not, the screen displays an error message and provides a chance to correct the errors.
Remove	Removes the display of the repository in Cisco Policy Builder. **Note** The remove link here does not delete any data at that URL. The local directory is deleted.

b. Click **OK** to save your work to the local directory.

Note When you change screens, the Policy Builder automatically saves your work. We recommend saving your work to the local directory by clicking on the diskette icon on the Policy Builder GUI or CTRL-S on the keyboard.

c. If you are ready to commit these changes to the version control software, choose **File > Save to Client Repository** on the Policy Builder home screen.

2. Log in to the new repository.
3. Verify the new CRD table schema and publish the changes.

4. Review the crd-api pod logs for any exception or error related to the duplicate key or duplicate index. If there are no errors, then the CRD is successfully imported.

Service barring configuration

Service Barring feature allows you to bar subscribers from a particular PIN code.

cnAAA supports:

- Service barring applies to new sessions based on Circuit ID, area, or PIN code.
- Service barring also applies to existing sessions based on Circuit ID, area, or PIN code.

Configuring the service barring functionality involves these tasks:

- Configure the CRD table
- Configure service template in the Policy Builder
- Configure the barring table and exclusion table entries in Control Center

Configure the CRD table

Use the customer reference table to configure exclusion and barring details. You can exclude barring based on a combination of the `circuit_id`, `subscriber`, `PIN code`, or `service type` attributes. The system applies barring only when the `Disable_Barring` parameter is set to `false`.

Perform these actions to configure the CRD table:

Procedure

Step 1 To configure the **Exclusion_table**

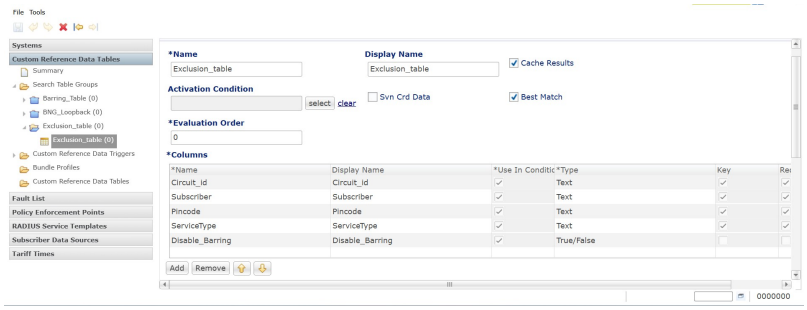
a) In the Policy Builder, navigate to the **Custom Reference Data Tables > Search Table Groups > Exclusion_table > Exclusion_table.**

b) Set the **Evaluation Order** to zero to make the table's output the activation condition for the **barring_table**.

Define these columns in the exclusion table:

- Input: Specify the `circuit_id` and other applicable fields.
- Output: Specify `Disable_barring value` as True/False.

Figure 14: Exclusion table



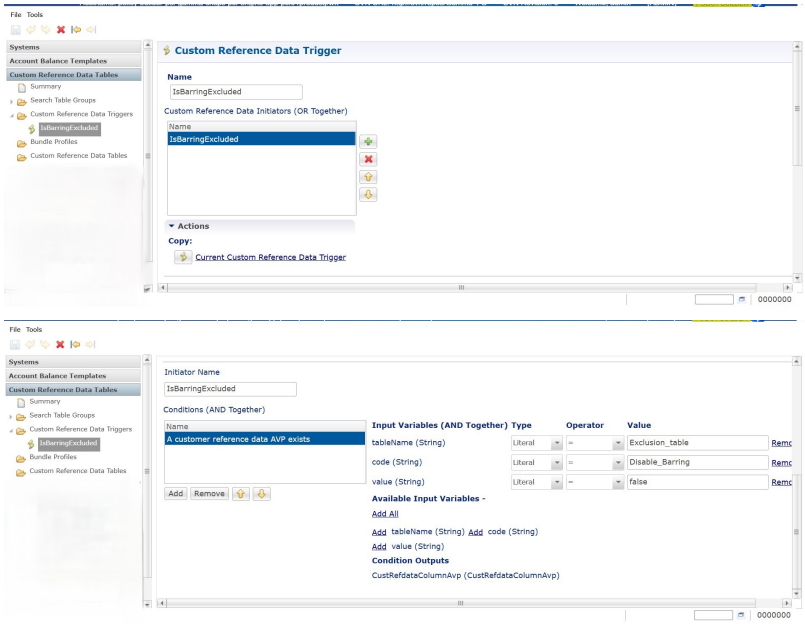
c) **disable_barring** is the result of the CRD evaluation.

- If **disable_barring** is set to True, the **barring_table** does not get evaluated.
- If **disable_barring** is set to `False`, the system applies barring to the matching subscribers.

Step 2 Set the activation condition for Barring:

a) Navigate to **Custom Reference Data Tables > Custom Reference Data Triggers > IsBarringExcluded**

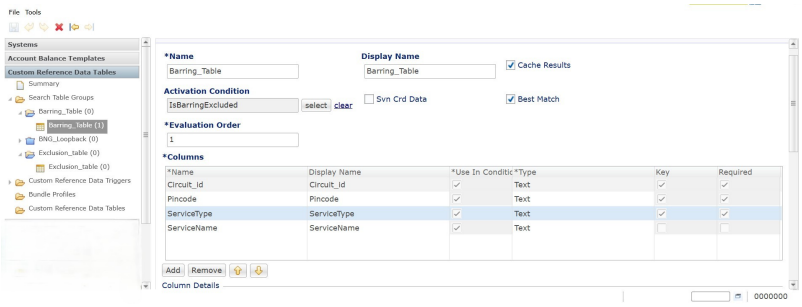
Figure 15: Call barring triggers



Step 3 To configure the **Generic Barring Table**:

a. Apply Barring based on the combination of attributes Circuit_id, pincode, ServiceType, Circuit_id Attribute

b. If required choose the **Radius Called Station Id Retriever** AVP from the list of input AVP's of CRD.



Note The ServiceName Attribute is the result of CRD evaluation.

Step 4 Configure the service template in the PB to associate each template with a static service name. If the service templates are not associated, the system uses input from the CRD output. If the CRD produces no result, the system considers the static input.

Configure service template in Policy Builder

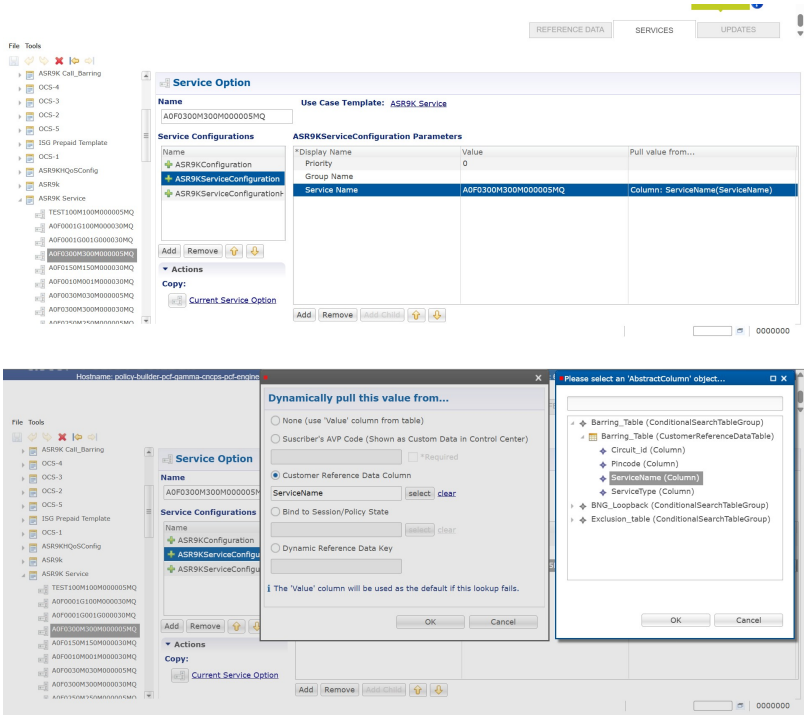
Follow these steps to configure the service template in Policy Builder:

Procedure

Step 1 Configure the service options for all High Speed Internet (HSI) services.

Step 2 Click the ellipsis (...) icon to dynamically retrieve the service name from the CRD table.

Figure 16: Mapping the service name from CRD



Note In the Policy Builder, choose **File > Publish to Runtime Environment** to publish the policy.

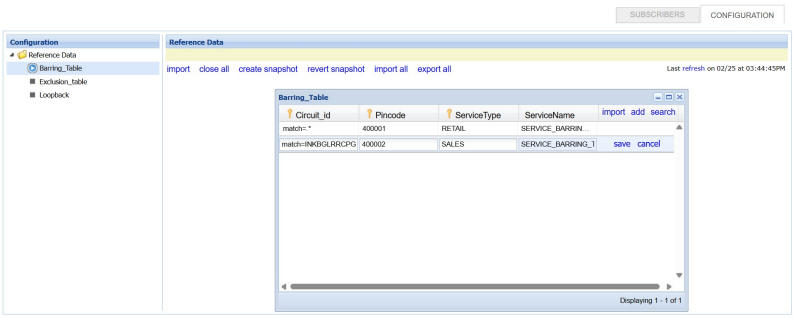
Configure the barring and exclusion table entries in Control Center

Procedure

Step 1 Login to the Content Center.

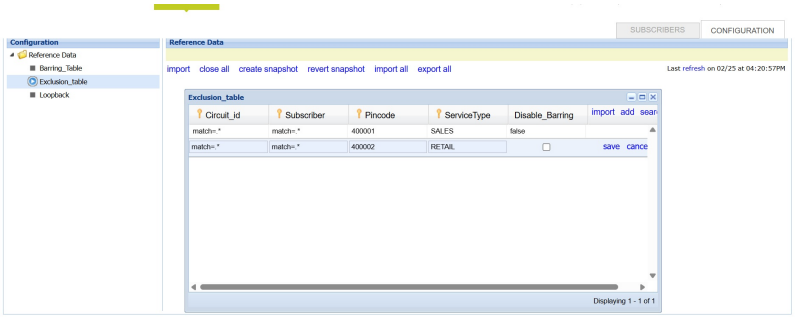
Step 2 Navigate to **Configuration**> **Barring Table**.

Step 3 In the **Reference Data** window, add entries to bar the subscribers.



Step 4 Navigate to **Configuration > Exclusion Table**.

Step 5 In the **Reference Data** window, **Add** entries to exclude subscribers from barring as shown in the example.



Step 6 Exclude the combination of ServiceType and Pincode from Barring.

Note You do not need to restart from the CLI or publish the policy after you add rows to the control center CRD tables.

You do not need to restart the cnAAA system through the Ops-Center CLI or publish the policy after adding rows to the CRD tables. The system automatically loads the CRD table entries.

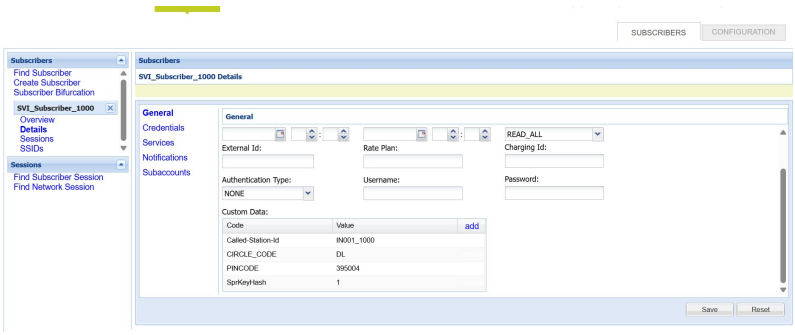
Sample SOAP request to provision subscriber PIN code AVP

You can include the PIN code and service type AVP for subscribers. These attributes serve as criteria for barring.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://broadhop.com/unifiedapi/soap/types">
<soapenv:Header/>
<soapenv:Body>
<subscriber>
<name> <fullName>EDRTest1</fullName> </name>
<credential>
<networkId>US4744_2</networkId>
<password>US4744_2</password>
</credential>
<service>
<code>A0F0005M005M000005MQ</code>
<enabled>true</enabled>
</service>
<status>ACTIVE</status>
<avp> <code>PINCODE</code> <value>400001</value> </avp>
<avp> <code>SERVICETYPE</code> <value>RETAIL</value> </avp>
</subscriber>
</soapenv:Body>
</soapenv:Envelope>
```

Verifying subscribers added:

View the provisioned subscriber details after SOAP Request.



Bar or unbar the services for existing sessions

Use the `call-barring.py` script to update the service plan for an existing session from the master node.

The script is located in the `/data/<namespace>/data-pcf-utilities-0/support/script/` directory.

```
cd /data/pcf-beta-cncps/data-pcf-utilities-0/support/script/
```

Execute the script using the `python3` command with the required arguments for PIN code, service name, called station ID, and SOAP URL.

```
python3 call-barring.py --pincode 395004 --servicename A0F0100M100M000030MQ --calledstationid
 0005.9A3C.7A00 --port 27017 --soapurl https://pcf.unified-api.192.0.2.1.example.com/ua/soap/
 --host mongo-admin-0
```

Example output:

```
luser@rid7508018-agandla-1-master1:~/test$ python3 callbarring.py --
pincode 395004 --servicename A0F0100M100M000030MQ --calledstationid
0005.9A3C.7A00 --port 27017 --soapurl https://pcf.unified-
api.10.127.34.151.nip.io/ua/soap/ --host mongo-admin-0
ARG servicename : A0F0100M100M000030MQ
```

```
ARG pincode : 395004
ARG calledstationid : 0005.9A3C.7A00
ARG Service Type :
ARG port spr mongoPort : 27017
ARG host spr mongoHost : mongo-admin-0
ARG Soap URL : https://pcf.unified-
api.10.127.34.151.nip.io/ua/soap/
Mongo Command: db = db.getSiblingDB("spr");
db.subscriber.find({"avps_key":{"$elemMatch":{"value_key": "395004",
"code_key": "PINCODE"}}}, {"_id": 0, "credentials_key.network_id_key": 1,
"services_key": 1}).forEach(printjson)
Namespace found: pcf
Kubectl command: kubectl exec -n pcf db-admin-0 -- mongo --port 27017 --host
mongo-admin-0 --eval db = db.getSiblingDB("spr");
db.subscriber.find({"avps_key":{"$elemMatch":{"value_key": "395004",
"code_key": "PINCODE"}}}, {"_id": 0, "credentials_key.network_id_key": 1,
"services_key": 1}).forEach(printjson)
Processing subscriber :0005.9A3C.7A00 for checking A0F0100M100M000030MQ,
Called Station id:0005.9A3C.7A00
Processed 1 records.
0 : Subscriber 0005.9A3C.7A00 is having the service :A0F0100M100M000030MQ
Response from server >> <Response [200]>
0 : Subscriber [0005.9A3C.7A00] SwitchService Was Success for Service
[A0F0100M100M000030MQ]
*** Done
luser@rid7508018-agandla-1-master1:~/test$
```

Import the new CRD table

This section describes how to import the CRD table.

To import new CRD tables, use the following configuration:

Before importing the CRD table, ensure that the CRD data archive is saved as dot (.) crd or dot (.) zip.

1. Log in to the cnAAA Central.
2. Click **Custom Reference Data**.
3. Click **Import/Export CRD Data**.
4. Under **Export Custom Reference Data**, the following options are displayed:
   - Select the **Use 'zip' file extension** check box to enable easier viewing of export contents for advanced users.
   - Select the **Export CRD to Golden Repository** check box to export CRD to golden repository which is used to restore cust_ref_data in case of error scenarios. A new input text box is displayed.
5. Add a valid SVN server hostname or IP address to push CRD to repository. You can add multiple hostnames or IP addresses by clicking on the plus sign.
6. Click **Export**.

Verify the successful export of CRD Table to golden repository

To verify of the export of the custom CRD table to the golden repository is successful, use the following configuration:

1. Log in to the cnAAA 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 the **Import** button to import the CRD data.
6. On successful import, verify that you receive a "Data imported" message on the cnAAA Central GUI.
7. Review crd-api pod logs for any exception or error related to duplicate key or duplicate index. If there are no errors, then the CRD is successfully imported.

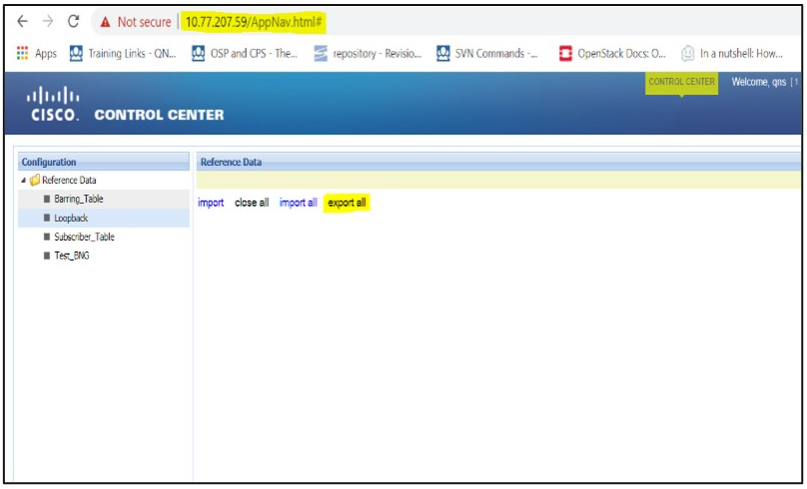
Bulk update for CRD table

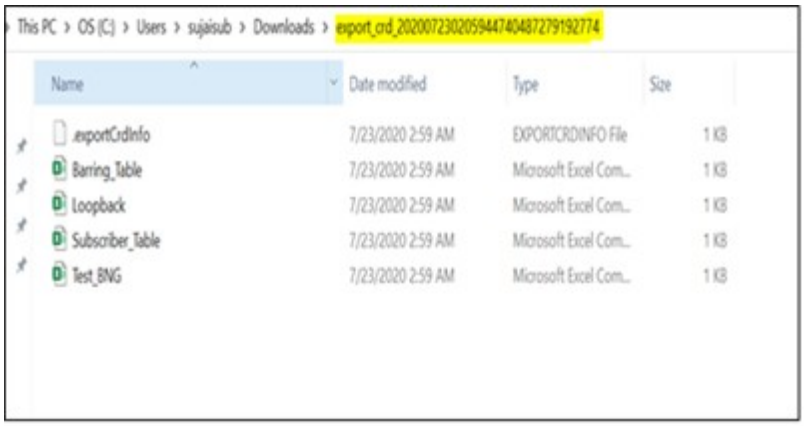
This feature enables you to perform bulk updates on Custom Reference Data (CRD) table records. Previously, CRD tables lacked **Export All** and **Import All** options, making it difficult to manage large volumes of data.

To perform a bulk update, complete these two phases:

Export and modify data

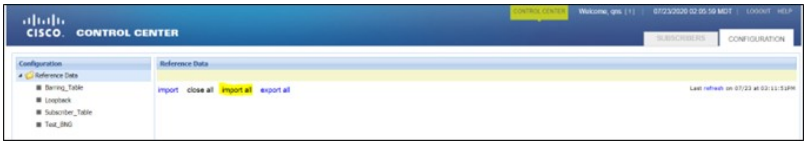
- Export the CRD table contents to a `.crd` file and unzip the `.crd` file to view and edit the table content in `.csv` files.
- The exported contents include the `.exportCrdInfo` file and the CSV tables.

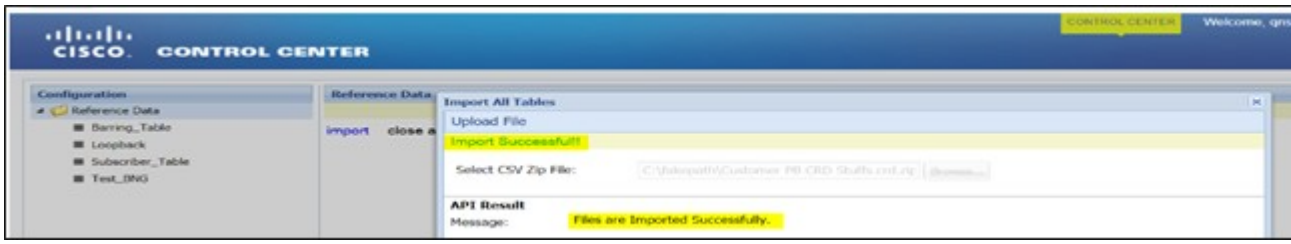




Import updated data

Import the updated `.csv` file contents back to the CRD table by zipping them into a `.zip` format.





Caution Imported contents overwrite the existing contents in the CRD tables.

Troubleshoot bulk updates

If you face issues while performing a bulk update, you can:

- Check the GUI error messages for issues with **Import All** or **Export All**.
- Refer to the `consolidated-qns.log` file in the `pcrfclient` VM.

Note Enable additional `DEBUG` level messages only in lab environments.

Figure 17: Export logs

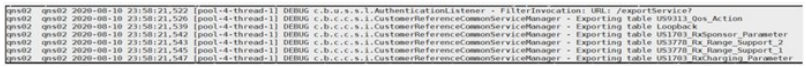
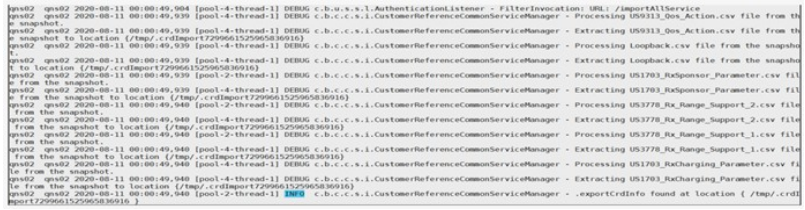


Figure 18: Import logs



CRD REST API for OLT NEID and loopback addition

CRD REST API (Add/Update/Delete/Query)

In custref data REST API service, provide Logging for audit purpose: For Create/Update/Delete Requests, logging is added for SourceIp Address, complete request for audit purpose.

The headers are uniform for all the functions of the API.

Configure CRD REST API (Add/Update/Delete/Query)

To perform various actions, follow the steps:

Procedure

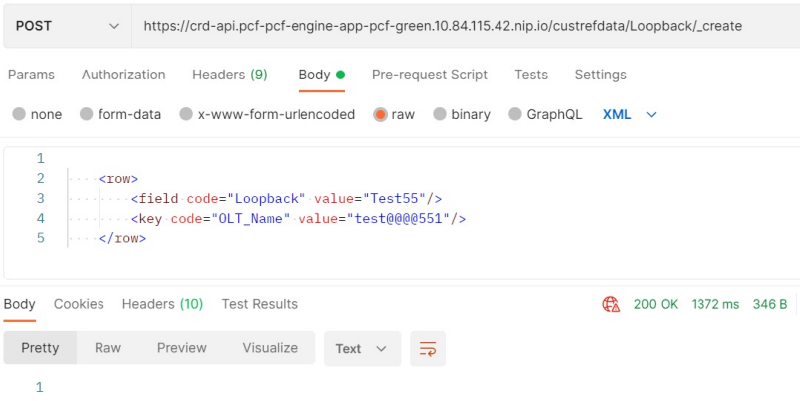
Step 1 Use HTTP headers to automate tasks.

Content-Type	application/xml
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive

Step 2 Add CustRefData loopback.

URL: https://crd-api.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/custrefdata/Loopback/_create

Output:

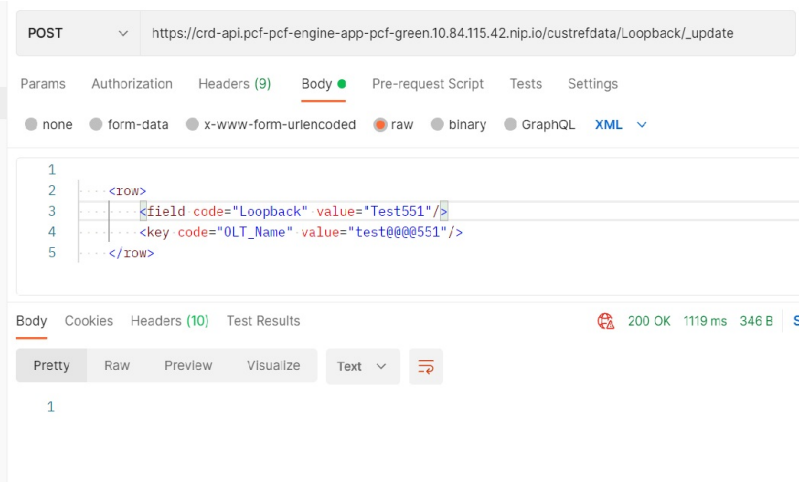


Step 3 Update CustRefData loopback.

URL: https://crd-api.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/custrefdata/Loopback/_update

Note Make sure the OLT name is accurate. Update the value after confirming its accuracy.

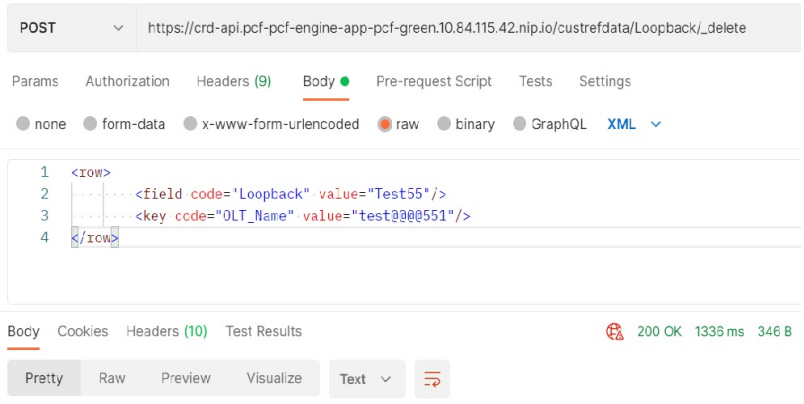
Output:



Step 4 Delete CustRefData loopback.

URL: https://crd-api.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/custrefdata/Loopback/_delete

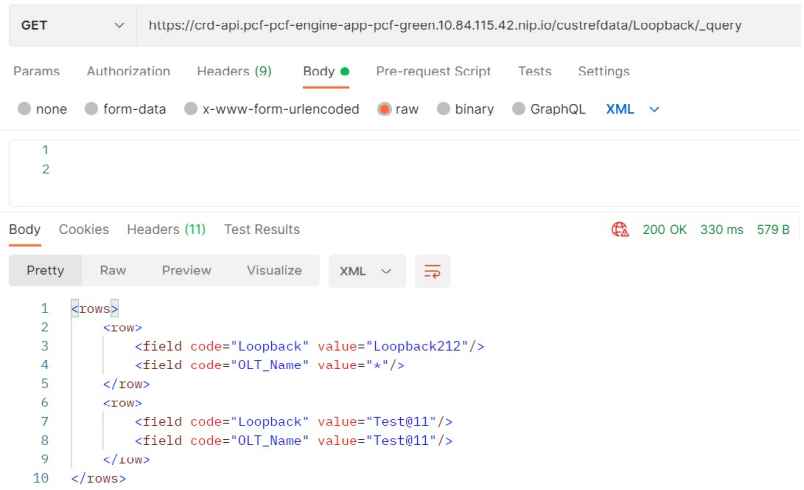
Output:



Step 5 Retrieve CustRefData loopback.

URL: https://crd-api.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/custrefdata/Loopback/_query

Output:



CRD refresh interval configuration and performance improvements in CRD functionality

Feature History

Feature Name	Release Information	Description
Customer reference data refresh interval configuration	2025.03.0	The CPC Performance and Scalability Enhancements optimizes CRD handling and enables Admin DB sharing. It allows configurable CRD cache refresh and secondary DB read preference to reduce primary load, while implementing Admin DB replica sets with specific IPs for cross-cluster reachability and simplified management.

Overview

The CPC Performance and Scalability Enhancements introduces two key improvements:

- Configurable CRD cache refresh intervals (in milliseconds) and the option to direct CRD cache rebuilds to secondary databases, which reduces the load on the primary database and ensures optimal CRD versioning.
- By default, the DB Read Preference is primary, but it is configurable in the CRD plugin configurations of PB to enable the CRD cache rebuilt using Secondary database rather than the Primary database to reduce the load on the Primary and to alleviate the overall system performance.
- Admin DB replica sets within the CPC namespace, configured with specific IP addresses instead of local defaults.

Previously there was no option to configure DB preference and CRD refresh interval, to improve the system performance

Configure CRD refresh interval with Ops-Center

Follow these steps to configure the Ops-Center engine parameters and the CRD refresh interval:

Procedure

Step 1 Enter the engine configuration mode for cpc-green.

```
engine <engine-name>
```

Step 2 Configure CRD Cache Refresh Interval: Set the desired refresh interval (in milliseconds) using two separate Ops-Center commands.

- crdapi crd-mongo-cache-refresh-interval value `<value_in_ms>` and commit the change.
- properties crd.mongo.cache.refresh.interval value `<value_in_ms>` and commit the change.



Note The interval values for `crdapi` and `crd.mongo` should be the same.

Step 3 Enter the exit command to complete and exit the configuration mode.

This sample configuration shows the commands for the Ops-Center engine:

```
engine cpc-green
crdapi crd-mongo-cache-refresh-interval 10000
properties crd.mongo.cache.refresh.interval
value 10000
exit
```

Step 4 Enable the debug logs in the Ops-Center with this configuration:

a) Set the default logging level for all loggers to "error" to minimize unnecessary logs.

```
debug logging default-level error
```

b) Configure debug level for CRD Manager logger.

```
debug logging logger com.broadhop.custrefdata.impl.CustomerReferenceDataManager
level debug
exit
```

c) Configure debug level for GenericDao logger.

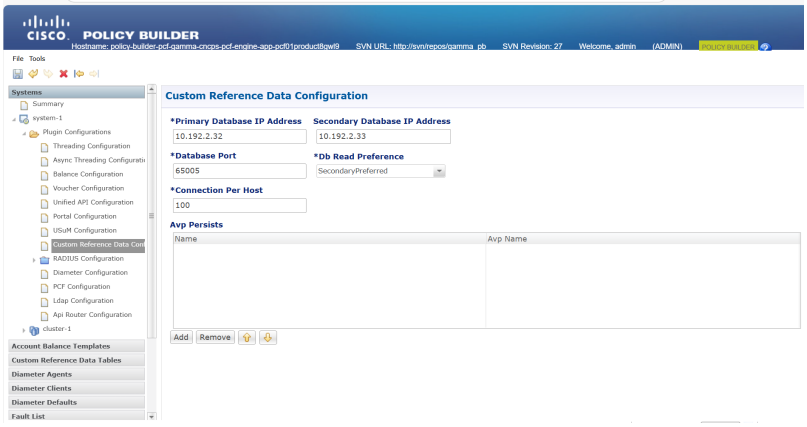
```
debug logging logger com.broadhop.custrefdata.impl.dao.GenericDao
level debug
exit
```

Example

```
debug logging default-level error
debug logging logger com.broadhop.custrefdata.impl.CustomerReferenceDataManager
 level debug
exit
debug logging logger com.broadhop.custrefdata.impl.dao.GenericDao
 level debug
exit
```

Configure DB read preference in Policy Builder

To optimize CRD cache performance and manage database load, configure the DB Read Preference directly within the Policy Builder interface:



Procedure

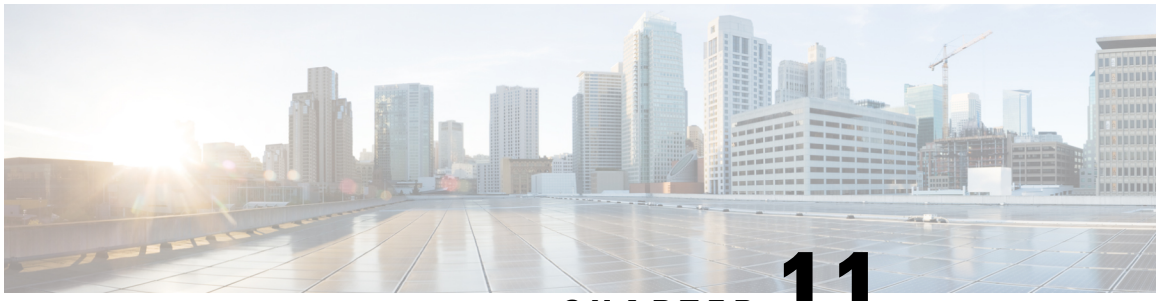
Step 1 In Policy Builder, navigate to **Plugin Configurations** under **Systems**.

Step 2 Select **Custom Reference Data Configuration**.

Step 3 Go to **DB Read Preference** drop down and select **SecondaryPreferred** to enable CRD cache rebuilds using a secondary database, which helps reduce the load on the primary database.

Note The read preference can only be pointed to the secondary when the multiple admin DB replica sets are configured in the CPC Ops-Center. It does not work with local DBs [admin-db-0, admin-db-1]

To ensure that changes take effect after modifying the DB Read Preference in Policy Builder, it is necessary to manually restart the Engine, CRD, Policy Builder pod, and Control Center pods.



CHAPTER 11

Message Prioritization and Overload Handling

- Overview, on page 217
- How it Works, on page 217
- Configure inbound message overload handling, on page 218
- Overload protection for RADIUS endpoint, on page 220
- OAM Support, on page 223

Overview

This chapter describes message prioritization and overload handling for RADIUS and cnAAA configurations. It details how to configure inbound message overload handling within Policy Builder and explains Message Handling Rules, which allow rules based on request type and priority. The chapter also provides information about bulk statistics for monitoring message prioritization and overload handling.

How it Works

This section describes how the message prioritization handling framework functions.

- Message Prioritization Handling Framework
  - RADIUS Configuration—Use Message Handling Rules parameters (RADIUS Client, Protocol, Command Code, Request Type, Priority, Per Instance TPS, and Discard Behavior) to identify and prioritize the RADIUS messages.
  - cnAAA Configuration—Use Message Handling Rules parameters (Request Type, Priority, Per Instance TPS, and Discard Behavior) to identify and prioritize the SBI messages.



Note Currently, the priority queue and rate limiting for REST and RADIUS messages are independent of each other.

Configure inbound message overload handling

This section describes how to configure the Inbound Message Overload Handling for the RADIUS and cnAAA configurations.

RADIUS Configuration

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 **RADIUS Configuration**.
6. In the right pane, to add the parameters of the inbound message overload handling, check the **Inbound Message Overload Handling** check box.
7. In the Inbound Message Overload Handling area, define the following parameter details.

Table 23: Inbound Message Overload Handling Parameters

Parameter	Description
Default Priority	Default priority to be assigned to an incoming message if no specific priority is defined in the Message Handling Rules table. Default value is 0.
Message Sla Ms	Service Level Agreement (SLA) in milliseconds, defines the number of milliseconds that are associated with an incoming event or message. In case the configured duration times out, the Discard Behavior configured in the Message Handling Rules is applied else the Default Discard Behavior is used. Maximum time (in millisec) that a message has in an inbound message handling queue waiting for a worker thread. Configuring this value avoids processing a message to time out by a remote peer. Default value is 1500 ms.
Inbound Message Queue Size	Allows the maximum number of messages in the Inbound Message Queue. When the number of messages exceeds this value, messages are discarded as defined in the Message Handling Rules and the Default Discard Behavior. Default value is 1000.

Parameter	Description
Default Instance Rate Limit	This parameter is applied to messages that do not have an applicable overload handling rule configured in the Message Handling Rules table. Default value is 0.
Default Discard Behaviour	Default behavior to be applied to an incoming message if no specific discard behavior for that message is defined in the Message Handling Rules table. • MESSAGE_DROP: Discards the request. Default value is MESSAGE_DROP
Message Handling Rules	Defines specific inbound message overload handling rules based on different criteria. For more information, see Table 24: Message Handling Rules Parameters, on page 219.

Figure 19: Inbound Message Overload Handling Parameters

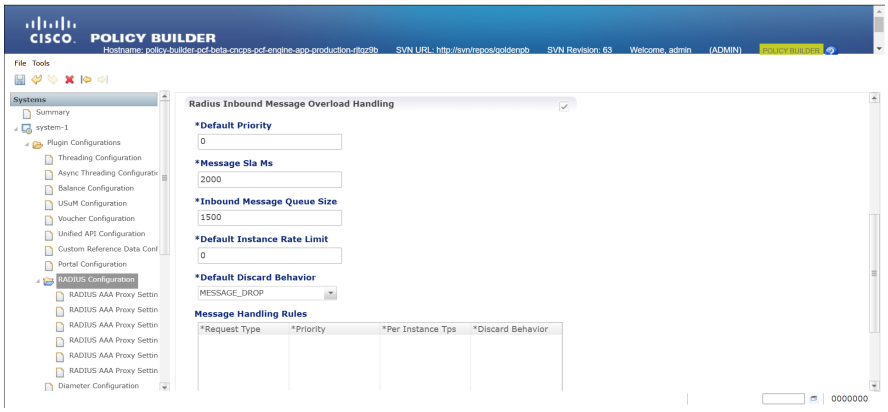


Table 24: Message Handling Rules Parameters

Parameter	Description
Request Type	Specific request type value to be used for scoring. AccessRequest SessionAccountingStart ServiceAcctInterimUpdate SessionAccountingStop ServiceAccountingStart ServiceAccountingStop

Parameter	Description
Priority	Priority value assigned to the message. Higher numerical value has the higher priority. Default value is 0. For example, 10, 20, 100, 200, 300, 500 and so on.
Per Instance Tps	Transactions per second limit per process. This value is the TPS that these messages are limited to. The actual system's transaction per second limit can be calculated using the following formula: Per Instance Tps x Number of instances per VM x Number of VMs. Default value is 0. For example, 1000, 2000, 5000 and so on.
Discard Behavior	Behavior to be applied to an incoming message. • MESSAGE_DROP: Discards the request. Default value is MESSAGE_DROP.

Overload protection for RADIUS endpoint

Feature History

Feature name	Release information	Description
Overload protection global for RADIUS end point	2025.03.0	This feature provides global protection to the Radius Endpoint. It allows to configure rate limits globally to avoid overload on the CPS system during high traffic scenarios.
Message prioritization and overload handling	2025.01.0	This feature provides configurable message prioritization and overload handling for RADIUS endpoints. It allows defining inbound controls for RADIUS authentication and accounting requests within Ops Center. This prevents message overload, ensuring RADIUS service stability and optimal performance.

Overview

The feature provides overload protection for global and individual message-level protection to the Radius Endpoint. This protects the cnAAA application from sudden, unpredictable load of Authentication and Accounting request messages. By managing incoming traffic, it prevents service degradation and ensures the continuous availability of RADIUS services.

Global overload protection in the RADIUS endpoint

Global overload protection manages incoming RADIUS messages in these stages:

1. **Global limit enforcement**: The system checks incoming RADIUS messages against the `max-requests-per-sec` global rate limit.
2. **Action on exceeding limit**: If the number of requests exceeds the configured maximum, the system drops the message.



Note Service accounting start and service accounting stop messages are excluded from the overload control limits.

3. **Message-level configuration**: If the global rate limit is not exceeded, the system evaluates the message based on message-specific configuration.
4. **Global limit precedence**: The global limit takes precedence as the primary control to maintain stability.

Configure RADIUS endpoint message rate limits

To configure message prioritization of RADIUS endpoint, follow these steps:

Procedure

Step 1 Log into the Ops-Center and use the `config` command to enter the message level configuration mode.

Step 2 Use the `throttle-action parameter` to configure the action on authentication message overload detection:

```
advance-tuning overload-control radius authentication action throttle-action DROP
```

Step 3 Use the `max-requests-per-sec` parameter to define the maximum number of RADIUS messages.

```
advance-tuning overload-control radius authentication limits max-requests-per-sec 1000
```

Step 4 Use the `throttle-action parameter` to configure the action on accounting message overload detection:

```
advance-tuning overload-control radius accounting action throttle-action DROP
```

Step 5 Use the `max-requests-per-sec` parameter to define the maximum number of RADIUS messages.

```
advance-tuning overload-control radius accounting limits max-requests-per-sec 1000
```

Note Service accounting start and service accounting stop messages are excluded from `max-requests-per-sec` limits.

Configure the message rate limit for global protection of RADIUS endpoint

Configure these parameters in the Ops Center to manage the global rate limiting behavior for RADIUS messages.

Procedure

Step 1 Log into the Ops Center and use the `config` command to enter the global configuration mode.

Step 2 Use the `max-requests-per-sec` parameter to define the maximum number of RADIUS messages.

```
advance-tuning overload-control radius global limits max-requests-per-sec <value>
```

Step 3 Use the `throttle-action parameter` command to define the action to take when the global rate limit is exceeded.

```
advance-tuning overload-control radius global action throttle-action DROP
```

This smaple configuration shows the global overload protection in Ops Center:

```
global :-
advance-tuning overload-control radius global limits max-requests-per-sec 1000
advance-tuning overload-control radius global action throttle-action DROP
```

Note Guava's RateLimiter may occasionally reject requests before reaching the configured transactions-per-second (TPS) limit due to its token bucket algorithm. To address this, use the reserve-percentage parameter to increase the effective rate limit, providing a buffer to ensure smoother throttling.

For example, if TPS of 500 and a reserve-percentage of 20%, the RateLimiter's internal limit is adjusted to 600 TPS (500 + 20%). This allows the limiter to accommodate slight variations in request patterns.Tune the RateLimiter based on this adjusted limit, noting that a small amount of traffic beyond the target TPS may be permitted, depending on the tuning.

Use this parameter to tune the threshold limit:

```
advance-tuning overload-control radius global limits reserved-percentage 20
```

CoA throttle support

cnAAA supports a configurable throttling mechanism that limits the CoA TPS based on the maximum number of CoA TPS permitted per BNG. Discuss with BNG team to confirm the throttling limit (value).

PB Configurations

The `throttling-limit` parameter, located under the RADIUS configuration in the Ops-Center, defines the throttling limit for ASR9K Policy Enforcement Points (PEPs).

When throttling for CoA is required, set the CoA Throttling for ASR9K PEP to true, configure the appropriate limit, and commit the changes.

```
radius advance-tuning throttling-limit 10
radius advance-tuning coa-throttlling-for-asr9k-pep true
```

This property specifies the minimum allowed throttle limit. If you set this property to 5, ensure the `throttle-limit` is not less than 5.

```
radius properties com.broadhop.minimum.throttling.limit value 5
```



Note

- Throttling functionality is applicable only for ASR9K devices.
- If you set the throttling limit to 10, each RADIUS endpoint pod processes up to 10 requests per second for each BNG. When the rate exceeds 10 transactions per second (TPS), the system forwards the CoA request to the engine for later processing.

OAM Support

This section describes operations, administration, and maintenance information for this feature.

Bulk Statistics support

The following statistics are supported for the message prioritization and overload handling feature.



Note The following values apply to all the statistics:

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

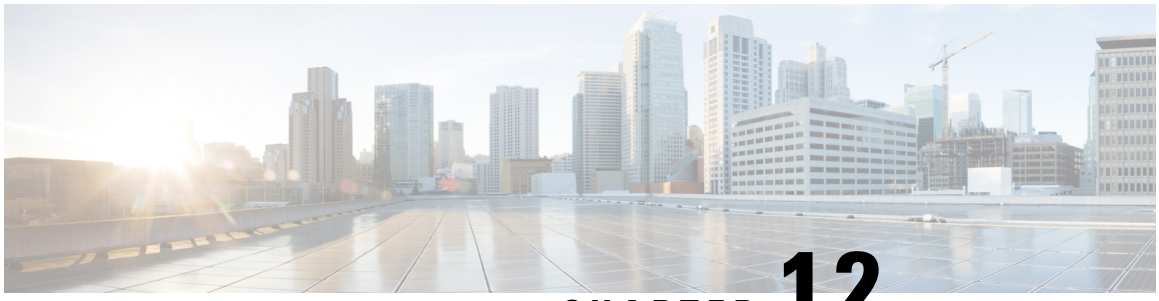
The following metrics track the counter information:

- input_queue_result - Captures the status of the message in the inbound queue whether it is dropped or rate limited.

  The following labels are defined for this metric:

  - appid
  - message-type

- result



CHAPTER 12

RADIUS Peer Load Rebalancing

- Feature Summary and Revision History, on page 225
- Feature Description, on page 225
- ULB integration on RADIUS Endpoint, on page 226

Feature Summary and Revision History

Summary Data

Table 25: Summary Data

Applicable Product(s) or Functional Area	cnAAA
Applicable Platform(s)	SMI
Feature Default Setting	Disabled – Configuration required to enable
Related Documentation	Not Applicable

Feature History

Table 26: Feature History

Feature Details	Release
First introduced.	2025.01.0

Feature Description

cnAAA supports RADIUS peer load rebalancing.

ULB integration on RADIUS Endpoint

Feature History

Feature Name	Release Information	Description
ULB Integration on RADIUS Endpoint	2025.01.0	The Unified Load Balancer enhances network performance by distributing traffic across endpoints while preserving client IP addresses for precise traffic management. It supports direct server-to-client responses and unique service IPs, ensuring low latency and reliable routing.

Overview

The Unified Load Balancer (ULB) improves network performance by distributing traffic across endpoints and preserving client IP addresses for accurate traffic management. It facilitates direct server-to-client responses and supports unique service IPs, ensuring low latency and reliable routing.

Prerequisites

Ensure the system is deployed with Container Network Interface (CNI) as Cilium and configured with the following Cilium settings:

```
[m13-cm] SMI Cluster Deployer# show running-config clusters | include cili
 configuration cni type cilium
 configuration cilium lb-algorithm random
 configuration cilium enable-legacy-host-routing true
 configuration cilium enable-egress-gateway true
[m13-cm] SMI Cluster Deployer#
```

Install and configure ULB for cnAAA

1. Download and prepare the Offline build.
   a. Use this command in the server terminal to download the build:

      ```
      wget https://engci-maven-master.cisco.com/
      artifactory/mobile-cnat-charts-release/
      releng-builds/
      2025.02.0/
      ulb/2025.02.0.i35/
      ulb.2025.02.0.i35-offline/
      ulb.2025.02.0.i35.SSA.tgz --user pcf-deployer.gen --ask-password
      ```

   b. Extract the archive:

```
tar -zxvf ulb.2025.02.0.i35.SSA.tgz
```

c. Move the extracted file:

```
mv ulb.2025.02.0.i35.tar ../
```

2. Prepare the software repository configuration.

a. Calculate SHA256 checksum:

```
sha256sum ulb.2025.02.0.i35.tar
```

b. Configure the repository:

```
software cnf ulb.2025.02.0.i35
 url      http://10.84.16.209/releases/ulb/ulb.2025.02.0.i35.tar
 user     <username>
 password <password>
 sha256   0efcdab729f9408053d5d3c7deb64e619d3919b045e78d3e3fb76c82ca18d373
exit
```

3. Commit ULB configuration in CM SMI deployer.

a. Log in to the Deployer:

```
ssh admin@<ip address> -p 2024
password: <Enter Password>
```

b. Commit configuration:

```
software cnf ulb.2025.02.0.i35
 url      http://10.84.16.209/releases/ulb/ulb.2025.02.0.i35.tar
 user     <username>
 password <password>
 sha256   0efcdab729f9408053d5d3e3fb76c82ca18d373
exit
clusters m13-cnaaa
 ops-centers lbs m13
  repository-local        ulb.2025.02.0.i35
  sync-default-repository true
  netconf-ip              <ip address>
  netconf-port            2024
  ssh-ip                  <ip address>
  ssh-port                2025
  httproutes-hostname        <ip address>.nip.io
  initial-boot-parameters use-volume-claims false
  initial-boot-parameters first-boot-password <password>
  initial-boot-parameters auto-deploy true
  initial-boot-parameters single-node false
 exit
exit
```

4. Initiate cluster synchronization.

a. Run synchronization:

```
clusters m13-cnaaa actions sync run upgrade-strategy concurrent debug true
```

b. Monitor logs:

```
monitor sync-logs m13-cnaaa
```

5. Configure ULB Ops-Center on remote cluster

a. Log in to Ops-Center:

```
ssh admin@<ip address> -p 2024
password: <password>
```

b. Set logging levels and cilium parameters:

```
logging level error
logging name lbs-agent.egress-mgr.app level error
...
cilium bpf auth-map-max    4194303
...
commit
```

c. Verify deployment:

```
show system
```

Sample output:

```
[m13-cnaaa/m13] lbs# show system
Mon Oct  27 11:59:15.599 UTC+00:00
system uuid 38789063-e32e-492d-91ab-265e85e13187
system status deployed true
system status percent-ready 100.0
system ops-center repository
https://charts.2002-10-192-1--21.sslip.io/ulb.2025.04.0.i18
system ops-center-debug status false
system synch running true
system synch pending false
[m13-cnaaa/m13] lbs#
```

6. Verify ULB Setup

a. Verify Services:

```
kubectl get emp -n pcf-m13
kubectl get lb -n pcf-m13
kubectl describe emp -n pcf-m13 lbs-emp-radius-ep-0
kubectl describe lb -n pcf-m13 lbs-radius-account-ep
kubectl describe lb -n pcf-m13 lbs-radius-auth-ep
```

Configuration for ULB Ops-Center

Sample Configuration:

```
logging level error
logging name lbs-agent.egress-mgr.app level error
logging name lbs-agent.egress-mgr.calico level error
logging name lbs-agent.egress-mgr.egressmgr level error
logging name lbs-agent.egress-mgr.ipt-reconciler level error
logging name lbs-agent.egress-mgr.iptables level error
logging name lbs-agent.egress-mgr.iptables-oper level error
logging name lbs-agent.egress-mgr.lbs-iptables level error
logging name lbs-agent.egress-mgr.listener level error
logging name lbs_operator.app.app level error
logging name lbs_operator.app.empcrd level error
logging name lbs_operator.app.lbcrd level error
logging name lbs_operator.app.service level error
cilium bpf auth-map-max    4194303
cilium bpf fragments-map-max 65536
cilium bpf lb-affinity-map-max 4194304
cilium bpf lb-map-max      4194304
cilium bpf lb-rev-nat-map-max 4194304
cilium bpf lb-service-backend-map-max 4194303
```

```
cilium bpf lb-service-map-max 4194304
cilium bpf lb-source-range-map-max 4194304
cilium bpf neigh-global-max 4194304
cilium bpf node-map-max    4194303
cilium bpf policy-map-max  16383


cloud-user@m13-cnaaa-master-1:~$ kubectl get namespace
NAME              STATUS   AGE
cee-m13           Active   17d
default           Active   17d
istio-system      Active   17d
kube-node-lease   Active   17d
kube-public       Active   17d
kube-system       Active   17d
lbs-m13           Active   17d
nginx-httproutes     Active   17d
pcf-m13           Active   17d
registry          Active   17d
smi-certs         Active   17d
smi-ops-control   Active   17d
smi-vips          Active   17d
cloud-user@m13-cnaaa-master-1:~$

kubectl label namespaces pcf-m13 ulb-mutate-enable-service-link=enable
```

These are the example output of various EMP, LB, and Kubernetes services related to the ULB feature.

EMP Services

```
kubectl get emp -n pcf-m13
NAME                   AGE
lbs-emp-radius-ep-0   3d23h
lbs-emp-radius-ep-1   3d23h
lbs-emp-radius-ep-2   3d23h
lbs-emp-radius-ep-3   3d23h
lbs-emp-radius-ep-4   3d23h
lbs-emp-radius-ep-5   3d23h
cloud-user@m13-cnaaa-master-1:~$ kubectl get lb -n pcf-m13
NAME                    AGE
lbs-radius-account-ep   3d23h
lbs-radius-auth-ep      3d23h
```

LB Services

```
kubectl get lb -n pcf-m13
Name:         lbs-emp-radius-ep-0
Namespace:    pcf-m13
Labels:       app=cps-radius-ep
              app.kubernetes.io/managed-by=Helm
              chart=cps-radius-ep-0.6.43-dev-cpc-2025-01-0117-250113145837-79379ea
              component=cps-radius-ep
              heritage=Helm
              release=pcf-m13-cps-radius-ep
Annotations:  meta.helm.sh/release-name: pcf-m13-cps-radius-ep
              meta.helm.sh/release-namespace: pcf-m13
API Version:  lbs.cisco.com/v1alpha1
Kind:         EgressManagementPolicy
Metadata:
  Creation Timestamp:  2025-01-17T07:54:00Z
  Generation:          1
  Resource Version:    3134597
  UID:                 e5012e8d-a78d-4462-80b2-2f4357ff86c6
Spec:
```

```
  Egress I Ps:
    101.101.167.131
  Egress Port:
    Stateful:
      Pod Name:  radius-ep-0
      Port Ranges:
        42000...42999
  Protocol:  UDP
  Target:
    Selectors:
      Match Labels:
        Component:  cps-radius-ep
Status:
  Endpoints:  1/1
  Services:   1/1
Events:       <none>
```

EMP Example

```
kubectl describe emp -n pcf-m13 lbs-emp-radius-ep-0
Name:         lbs-radius-account-ep
Namespace:    pcf-m13
Labels:       app=cps-radius-ep
              app.kubernetes.io/managed-by=Helm
              chart=cps-radius-ep-0.6.43-dev-cpc-2025-01-0117-250113145837-79379ea
              component=cps-radius-ep
              heritage=Helm
              release=pcf-m13-cps-radius-ep
Annotations:  meta.helm.sh/release-name: pcf-m13-cps-radius-ep
              meta.helm.sh/release-namespace: pcf-m13
API Version:  lbs.cisco.com/v1alpha1
Kind:         LoadBalancer
Metadata:
  Creation Timestamp:  2025-01-17T07:54:00Z
  Generation:          1
  Resource Version:    3133122
  UID:                 a6c0f711-503f-4224-8a43-cfba7363f649
Spec:
  Protocol:  UDP
  Service I Ps:
    101.101.167.131
  Service Port:  1813
  Target:
    Port:  1813
    Selectors:
      Match Labels:
        Component:  cps-radius-ep
Status:
  Endpoints:  0/0
  Services:   1/1
Events:       <none>
```

Ops Center configuration to enable ULB on RADIUS Endpoint

To enable the ULB in cnAAA, configure the ops-center as follows:

1. Set the ULB parameter to "true" to activate the backend load balancing service.
2. If the ULB parameter is "false" the system uses default Kubernetes capabilities for traffic management.

General RADIUS Settings

```
cnaaa# show running-config radius | nomore

radius accounting-port 1812
radius authorization-port 1813
radius coa-port 2799

radius bind-ipv4 [ bind IPv4 address ]
radius bind-ipv6 [  bind IPv4 address ]
radius replicas 2
radius lbs-service true
radius settings request-timeout-ms 5000
radius settings max-tries 4
radius settings min-processing-time-millis 3000
radius settings backoff-time-millis 1000
```

Configuration Notes

- radius bind-ip host_address: Specifies the host address for the RADIUS binding.
- replicas number_of_replicas: Indicates the number of replicas.
- ULB-service boolean_value: Specifies whether the load balancing service is enabled (true or false).
- settings request-timeout-ms timeout_in_milliseconds: Sets the request timeout in milliseconds.
- settings max-tries max_attempts: Defines the maximum number of retry attempts.
- settings min-processing-time-millis min_processing_time_in_milliseconds: Sets the minimum processing time in milliseconds.
- settings backoff-time-millis backoff_time_in_milliseconds: Sets the backoff time in milliseconds.

RADIUS Device Group: ASR9K

```
radius device-group ASR9K

 default-shared-secret     sh512

 default-coa-shared-secret cisco

 device dev1

  ip                 10.1.2.12

  shared-secret  cisco

  coa-shared-secret  cisco

  loopback-addresses [ 12.3.1.2 ]

 exit

 device dev2

  ip                 3.4.5.6

  shared-secret      cisco

  coa-shared-secret  cisco

  loopback-addresses [ 3.4.8.1 ]

 exit
```

```
exit
```

Configuration Notes

- default-shared-secret cisco – Specifies the default shared secret for RADIUS communication.
- default-coa-shared- cisco – Specifies the default shared secret for CoA communication.
- device dev1 – Configures a RADIUS device with specific settings: ip– The IP address of the device.
- shared-secret cisco – The shared secret for communication with the device.
- coa-shared-secret cisco – The shared secret for CoA communication with the device.
- loopback-addresses – Specifies the loopback addresses for the device.

RADIUS Server Group: grp1

```
radius server-group grp1

 servers serv1

  primary              3.4.4.4

  secondary            5.5.5.3

  nas-ip               1.1.2.2

  accounting-port      8312

  authorization-port   1312

  auth-protocol        PAP

  radius-password      <password>

  shared-secret        <secret>
  timeout-seconds      20

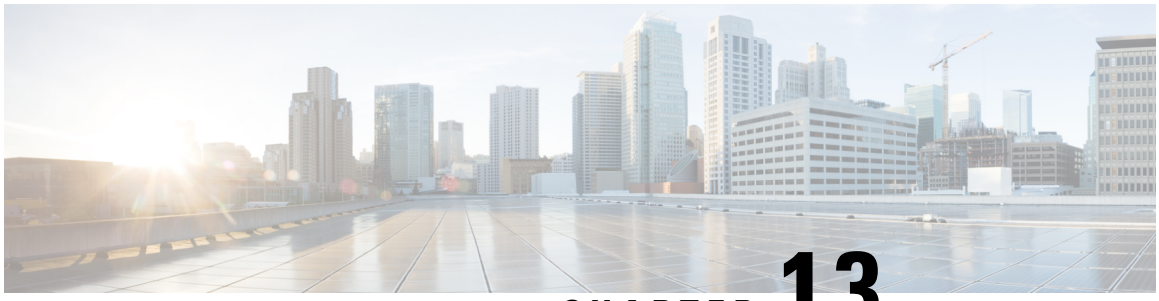
  test-message         false
  server-type   online (or) offline
```

Configuration Notes

- servers serv1 – Specifies the servers within the group.
- primary– The primary server's IP address.
- secondary – The secondary server's IP address.
- nas-ip– The IP address of the Network Access Server (NAS).
- authorization-port – The port used for authorization requests within the server group.
- auth-protocol PAP – Specifies the authentication protocol used, in this case, PAP.
- radius-password cisco– The password used for RADIUS authentication.
- shared-secret cisco – The shared secret used for communication within the server group.

- timeout-seconds 20 – Specifies the timeout duration in seconds for server responses.
- test-message false – Indicates whether test messages are enabled or disabled.
- test-userid cisco – The user ID used for test purposes.
- test-password cisco – The password used for test purposes.
- thread-pool-size 10 – Specifies the size of the thread pool for handling requests.
- max-proxy-queue-size 3 – Specifies the maximum size of the proxy queue.
- server-type - Specify as "online" for OCS server and "offline" for Passive-MZ server.
- radius properties grpc.executors – Configures properties related to gRPC executors.
- value 40 – Specifies the value for the gRPC executors property.

For more information on ULB, see the *Unified Load Balancer Configuration and Administration Guide*.



CHAPTER 13

Pods and Services

- Feature Summary and Revision History , on page 235
- Feature Description, on page 236
- Configuration Support for Pods and Services, on page 241
- Associate Pods to the Nodes, on page 241
- View the Pod details and status, on page 242
- States, on page 243
- Grafana dashboard files, on page 243

Feature Summary and Revision History

Summary Data

Table 27: Summary Data

Applicable Product(s) or Functional Area	cnAAA
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Always-on
Related Documentation	Not Applicable

Feature History

Table 28: Feature History

Feature Details	Release
First introduced.	2025.01.0

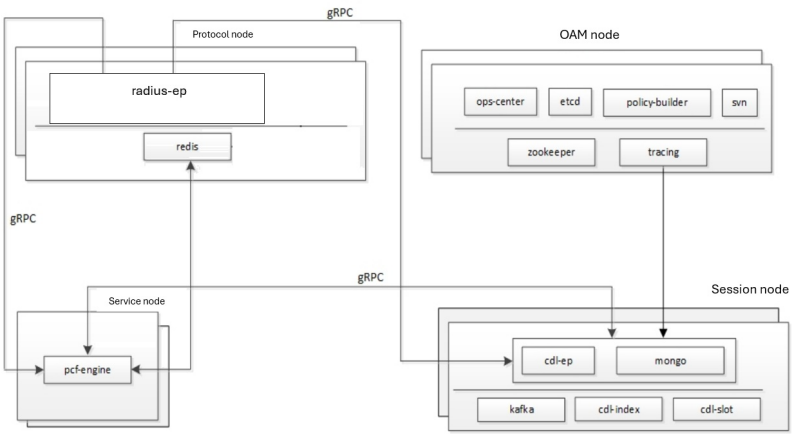
Feature Description

The cnAAA is built on the Kubernetes cluster strategy, which implies that it has adopted the native concepts of containerization, high availability, scalability, modularity, and ease of deployment. To achieve the benefits offered by Kubernetes, cnAAA uses the construct that includes the components such as pods and services.

Depending on your deployment environment, cnAAA 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 cnAAA spins new pods to replace the invalid pods.

The following 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 Node hosts the radius-ep, which manages httproutes (incoming) and egress (outgoing) traffic on the interfaces. Pods for operations and management processes are located in the OAM Node, while the Service Node hosts the cnAAA-engine. Session Nodes contain pods that serve as databases, storing data accessed by other pods. The illustration also shows the services that pods use to facilitate interactions. All pod communications occur over the gRPC interface.

Note To ensure resiliency, multiple instances of the Protocol and OAM Nodes are created

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.

For generic information on the Kubernetes concepts, see the Kubernetes documentation.

For more information on the Kubernetes components in cnAAA, see the following.

- Pods

- Services

Pods

Pod is a process that runs on your Kubernetes cluster. Pod encapsulates a granular unit known as a container. A pod can contain one or multiple containers.

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.

The following table lists the pod names and the hosts on which they are deployed depending on the labels that you assign. For information on how to assign the labels, see Associating Pods to the Nodes.

Table 29: cnAAA Pods

Pod Name	Description	Label
admin-db	Acts as the MongoDB router pod for the Admin database.	Session
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
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
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

Pod Name	Description	Label
rs-controller-spr1	Operates as a replication controller for the SPR database.	Session
cpc-engine-cpc-cpc-engine-app-cpc	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
smart-agent-cpc-ops-center	Operates as the utility pod for the cpc Ops Center.	OAM
svn	Stores all the cpc XMI configuration files.	OAM
swift-cpc-ops-center	Operates as the utility pod for the cpc Ops Center.	OAM
traceid-cpc-cpc-engine	Stores the subscriber tracing details.	OAM
zookeeper	Assigned for the Zookeeper.	OAM
api-cpc-ops-center	Functions as the confD API pod for the cnAAA Ops-Center.	OAM
consolidated-aaa-logging-0	Preserves the logs generated by other pods like engine pod.	OAM
cps-license-manager	Acts as the cnAAA License Manager.	OAM
documentation	Contains the documentation.	OAM
etcd-cpc-etcd-cluster	Hosts the etc-d for the cnAAA application.	OAM
grafana-dashboard-cdl	Contains the Grafana metrics for CDL.	OAM
grafana-dashboard-cpc	Contains the Grafana metrics for cnAAA.	OAM
network-query	Operates as the utility pod to determine the route IP for the RADIUS outbound messages.	OAM
ops-center-cnaaa-ops-center	Acts as the cnAAA Ops Center.	OAM
patch-server-cpc-cnat-cps-infrastructure	Operates as the utility pod for patching the cpc JAR files.	OAM
cpc-day0-config-cpc-cpc -engine-<*n*>-rchg2	Dedicated for performing the Day-0 configuration for cpc.	OAM
policy-builder-cpc-cpc-engine-app	Operates as the Policy Builder for cpc.	OAM
api-proxy-logging-0	This post contains all the log related to unified proxy ep for debugging and troubleshooting.	OAM

Pod Name	Description	Label
kafka	Hosts the Kafka details for the CDL replication.	Protocol
crd-api-cpc-cpc-engine-app-cpc-<*n*>-mjgxp	Hosts the CRD APIs.	Protocol
radius-ep-<N>	Contains the RADIUS stack details and acts as the endpoint. **Note**:Configuration changes done in the radius opscenter properties causes the radius ep pod to restart automatically.cisco recommends making such changes only within the maintainance window.	Protocol
redis-keystore	Operates as the REDIS Index.	Protocol
redis-queue	Operates as the REDIS IPC.	Protocol
unified-api-proxy-ep	Unified API request handler during migration stage.	Protocol

Table 30: ULB Pods

Pod Name	Description	Label
lbs-agent	A DaemonSet pod managing NAT and RAW table entries on each Kubernetes node for enforcing Egress Management Policies at the node level.	OAM
lbs-operator	A controller pod responsible for automating deployment, scaling, and operational tasks of Kubernetes and Cilium resources associated with Load Balancer and Egress Management Policy CRs.	OAM
ops-center-lbs-m13-ops-center	A central pod for managing configuration, upgrades, and lifecycle of ULB components within the Kubernetes cluster.	OAM

Services

The cnAAA configuration is composed of several microservices that run on a set of discrete pods. Microservices are deployed during the cnAAA deployment. cnAAA uses these services to enable communication between the pods. When interacting with another pod, the service identifies the pod's IP address to start the transaction and acts as an endpoint for the pod.

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

Table 31: cnAAA Services and Pods

Service Name	Pod Name	Description
admin-db	admin-db-0	Serves to process the MongoDB-specific router messages.
crd-api-pcf-pcf-engine-app-pcf	crd-api	Processes the CRD API calls.
datastore-notification-ep	pcf-engine	Responsible for sending the notifications from the CDL to the 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 cnAAA.
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 cnAAA Ops Center.
patch-server-cnAAA-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 cnAAA-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.

Service Name	Pod Name	Description
policy-builder-pcf-pcf-engine-app-pcf	policy-builder	Manages the Policy Builder's request/response messages.
cps-radius-ep service details.	cps-radius-ep-<N>	Acts as a RADIUS endpoint
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

This feature has the following limitations in this release:

When removing a node using the **kubectl drain** command, the pods managing the inbound traffic such as cnAAA-radius-ep, and cnAAA-server-ep 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.

Configuration Support for Pods and Services

This section describes how to associate pods to node and view the pod-related information using the following steps:

1. Associating Pods to the Nodes
2. Viewing the Pod Details and Status

Associate Pods to the Nodes

This section describes how to associate a pod to the node based on their labels.

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 the following configuration:

```
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
```

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#
```

NOTES:

- If you opt not to configure the labels, then cnAAA assumes the labels with the default key-value pair.
- **cdl-layer**—Configures the key-value pair parameters for the CDL.
- **oam-layer**—Configures the key-value pair parameters for the OAM 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.

View the Pod details and status

This section describes how to view the pod details.

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

You can run the kubectl command from the master node to manage the Kubernetes resources.

- To view the summary of pod details, use this configuration:

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

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

- To view the summary of the pod details, use the following configuration:

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

States

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

The following table describes the pod's states.

Table 32: 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	Indicates that all the containers in the pod are successfully terminated. These pods cannot be restarted.
Failed	One ore more containers in the pod have failed the termination process. The failure occurred as the container either exited with non zero status or the system terminated the container.
Unknown	The state of the pod could not be determined. Typically, this could be observed because the node where the pod resides was not reachable.

Grafana dashboard files

Use this reference to identify each supplied dashboard file and its monitoring area.

The JSON files provide separate cnAAA monitoring views in Grafana.

Prerequisites

Before you import the dashboards, ensure that these JSON files are available:

- `cnAAA_Radius_TPS-KPIs.json`: RADIUS TPS monitoring.
- `cnAAA_Pod_Level_CPU_Memory_Utilization_KPIs.json`: Pod-level CPU and memory utilization.

- `cnAAA_Node_Level_CPU_Memory_Utilization_KPIs.json`: Node-level CPU and memory utilization.
- `cnAAA_Resource_Utilization_KPIs.json`: CPU and memory monitoring.
- `cnAAA_Soap_API_KPIs.json`: SOAP API monitoring.
- `cnAAA_Soap-Proxy_CPC_Migration_KPIs.json`: SOAP Proxy and CPC migration monitoring.

For information about file availability or for assistance importing dashboards, contact the Cisco cnAAA engineering team at cnaaa-ft-svi-team@cisco.com.

cnAAA Grafana dashboards

The cnAAA organizes Grafana dashboards into separate, task-oriented views rather than a single consolidated dashboard. This reorganization improves scannability, enabling you to locate specific statistics quickly during production issues and maintenance.

Dashboard organization

Use your existing Grafana URL to open the dashboards. The split-view organization is intended to simplify navigation and reduce the time required to isolate performance data.

Available dashboards

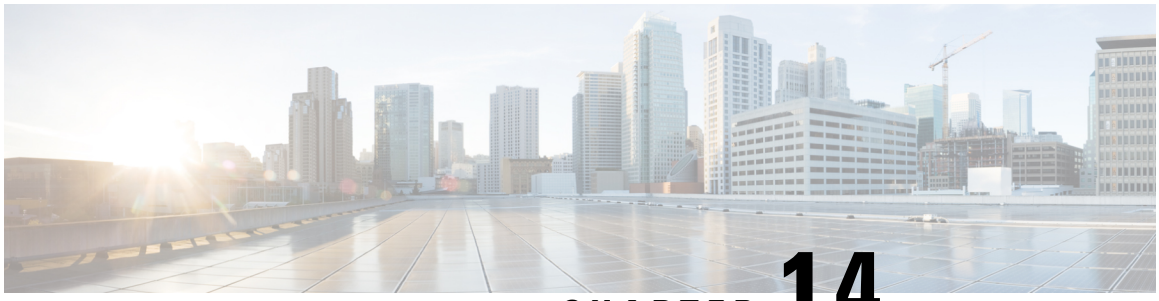
This table lists the available cnAAA Grafana dashboards and their monitoring areas.

Table 33: Grafana dashboard monitoring areas

Dashboard title	Monitoring area
cnAAA application monitoring – RADIUS TPS KPIs	RADIUS performance and throughput.
cnAAA pod level CPU memory utilization KPIs	Pod-level resource utilization.
cnAAA node level CPU memory utilization KPIs	Node-level resource utilization.
cnAAA application monitoring – SOAP API KPIs	SOAP API monitoring.
SOAP-Kafka migration dashboard (by source IP)	SOAP Proxy and CPC migration monitoring

Technical notes

- Data distribution: The dashboard split does not change the underlying traffic distribution.
- Dashboard deployment: The system does not include the JSON dashboard files in the software package. You must import these files separately.



CHAPTER 14

Advanced Tuning Parameters

- Feature Summary and Revision History, on page 245
- Feature Description, on page 246
- Configuration support for the Advanced Tuning parameters, on page 246
- Threading configuration for HTTP2 outgoing requests, on page 252
- Istio resource control configuration, on page 252
- Redis password configuration, on page 253
- Network Slice access control, on page 253
- OAM Support, on page 253
- Radius engine CPU configuration, on page 254
- KPI scheduler, on page 256

Feature Summary and Revision History

Summary Data

Table 34: Summary Data

Applicable Product(s) or Functional Area	cnAAA
Applicable Platform(s)	SMI
Feature Default Setting	Disabled – Configuration required to enable
Related Documentation	Not Applicable

Feature History

Table 35: Feature History

Feature Details	Release
First introduced.	2025.01.0

Feature Description

The cnAAA Ops Center allows you to configure the advanced tuning parameters for cnAAA. The tuning parameters primarily consist of the async-threading and http2-threading parameters. These parameters provide the flexibility of the tuning threads responsible for cnAAA's incoming and outgoing requests over HTTP.

Note Configure the advanced tuning parameter values only if you have a strong understanding of the cnAAA deployment.

cnAAA supports the message threshold per endpoint.

Note Message threshold is applicable only for the configured message types in RADIUS-endpoint

Configuration support for the Advanced Tuning parameters

This section describes how to configure the advanced tuning parameters using the CLI. The configuration of the advanced tuning parameters involves:

- Threading Configuration for HTTP2 Outgoing Requests
- Istio Resource Control Configuration
- Redis Password Configuration
- Network Slice Access Control

On or Off configuration to enable or disable the features

CPC introduces On/Off control for features to optimize system performance, enhance management, and align with operational needs by activating or deactivating specific functionalities.

How features are controlled

Features within CPC are managed through three methods:

1. **User Interface (UI) toggles**: For application-layer features accessible directly within the Policy Builder and Control Center interfaces, typically through check-boxes or dedicated settings.
2. **Ops-Center CLI**: For core engine or RADIUS based functionalities, managed by setting specific properties through the Ops-Center CLI.
3. **CRD based control**: For features activated or deactivated by executing specific automated scripts on demand.

Configure Ops-Center CLI

Procedure

Follow these steps to set properties through Ops-Center CLI:

Step 1 Obtain the Ops-Center CLI IP Address.

```
kubectl get svc -A | grep ops-center-<your-cluster-name>
```

Step 2 SSH into the **Ops-Center CLI**.

```
ssh -p 2024 admin@<Ops-Center-IP>
```

Step 3 Enter Configuration Mode.

```
[unknown] pcf# config
```

Step 4 Configure the property.

```
[unknown] pcf(config)# engine <engine-group-name> properties
 properties <property-key>
  value <true/false or string>
 exit
```

Step 5 Commit the Changes.

Example

```
cloud-user@m13-cnaaa-master-3:~$ kubectl get svc -A| grep ops-center-pcf-m13-ops-center
pcf-m13         netconf-ops-center-pcf-m13-ops-center                   ClusterIP
10.102.242.82         10.84.16.219
 2024/TCP                              23d
pcf-m13         ops-center-pcf-m13-ops-center                           ClusterIP
10.102.56.128         <none>
 8008/TCP,8080/TCP,2024/TCP,2022/TCP     23d
pcf-m13         ssh-ops-center-pcf-m13-ops-center                       ClusterIP
10.102.101.218        10.84.16.219
 22/TCP                                23d
cloud-user@m13-cnaaa-master-3:~$
cloud-user@m13-cnaaa-master-3:~$
cloud-user@m13-cnaaa-master-3:~$ ssh admin@10.102.56.128 -p 2024
admin@10.102.56.128's password:

      Welcome to the pcf CLI on m13-cnaaa/m13
      Copyright © 2016-2023, Cisco Systems, Inc.
      All rights reserved.

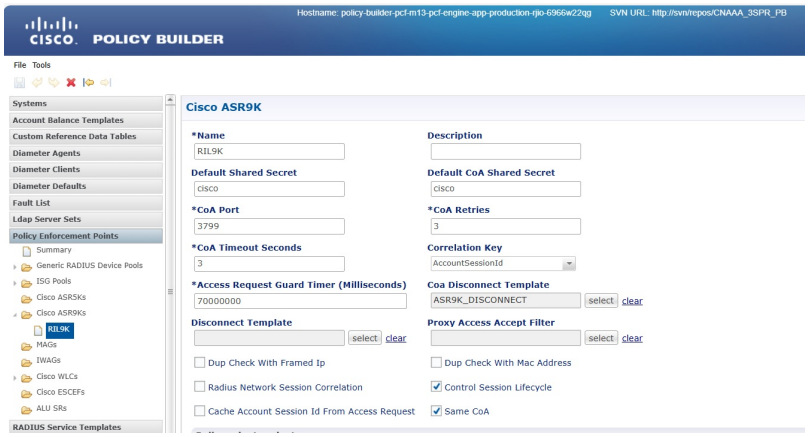
User admin last logged in 2025-10-23T08:45:30.882804+00:00, to
ops-center-pcf-m13-ops-center-95874d9b9-xwxpr, from 10.192.1.24 using cli-ssh
admin connected from 10.192.1.24 using ssh on ops-center-pcf-m13-ops-center-95874d9b9-xwxpr
[m13-cnaaa/m13] pcf#
```

List of features and controls

This section lists features that can be enabled or disabled, along with management instructions:

- **Combined multi CoA support**: This feature enhances CoA capabilities, allowing complex or simultaneous CoA operations. Policy Builder's UI manages this feature. It can be enabled by checking the dedicated checkbox in the PEP section. The default state is disabled.
- In Policy Builder's PEP section, check the `same COA` checkbox to activate the Combined Multi CoA feature.

Figure 21: Activate Combined Multiple CoA



- **Policy Builder and Control Center activity logging**: Logs various activities within PB and Control Center for auditing and monitoring. Control Center logs user login, logout, and subscriber-level CRUD operations to `qns-audit.log`. PB logs user login, logout, and policy publish events to `qns-audit-pb.log`. Setting properties through Ops-Center CLI:
  - Control Center logging:

    ```
    engine <engine group name> properties com.broadhop.cc.auditLog value true/false
    exit
    ```

  - Policy Builder Logging:

    ```
    engine <engine group name> policy-builder properties com.broadhop.pb.auditLog value
     true/false exit
    ```

Note This feature is enabled by default if the property is not configured in Ops-Center. To disable the feature, set the property value to `false`.

- **Policy Builder and Control Center login user details**: Records the most recent successful login event for each user in both Control Center and Policy Builder, including the username and exact timestamp. Control Center logs to `lastlogin_user_cc.log`, and Policy Builder logs to `lastlogin_user_pb.log`. OpsCenter Command Line Interface (CLI) manages this feature.
  - Center login details are managed by setting:

    ```
    engine <engine group name> properties com.broadhop.cc.login.details.feature value
    true/false exit
    ```

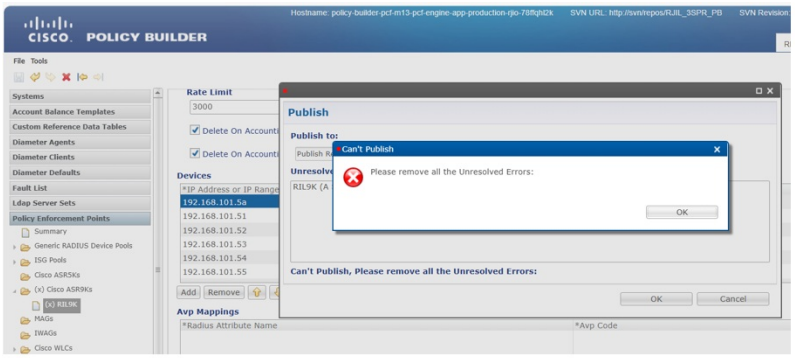
  - Policy Builder login details are managed by setting:

    ```
    engine <engine group name> policy-builder properties
    com.broadhop.pb.login.details.feature value true/false exit
    ```

Note This feature is enabled by default if the property is not configured in Ops-Center. To disable the feature, set the property value to `false`.

- **Unresolved errors display during publish**: Controls whether a pop-up dialog displaying configuration-related errors appears during Policy Builder publish operations. Disabling it prevents the dialog from showing.

Figure 22: Unresolved error




```
engine <engine group name> policy-builder properties com.broadhop.unresolvedError.feature
 value true/false exit
```

Note This feature is enabled by default if the property is not configured in Ops-Center. To disable the feature, set the property value to `false`.

- **Support all four IPv6 formats in PEP configuration**: Enables the system to recognize and process all four common IPv6 address formats within Policy and Accounting Traffic Steering (PATS) and PEP configurations.

```
engine <engine group name> properties com.broadhop.pep.ipv6.enable.feature value
<true/false> exit
engine <engine group name> policy-builder properties com.broadhop.pep.ipv6.enable.feature
 value <true/false> exit
```

Note This feature is enabled by default if the property is not configured in Ops-Center. To disable the feature, set the property value to `false`.

- **Support IPv6 address in domains**: Enhances domain selection during subscriber authentication by allowing matching of incoming Framed-IPv6-Address or NAS-IPv6-Address from RADIUS requests with IPv6 entries in a domain's locations section.

```
engine <engine group name> properties com.broadhop.domain.ipv6.enable.feature value
<true/false> exit
```

Note This feature is enabled by default if the property is not configured in Ops-Center. To disable the feature, set the property value to `false`.

- **CoA backoff retry on CoA timeout and CoA NACK from BNG**: Provides a configurable retry mechanism for CoA messages when a timeout occurs or a CoA-NACK is received from the BNG. The default state is enabled, with a default value of 300.

  ```
  radius properties backOffRetryCoA.maxRetransmission value <Set value: 0 to disable, >1
   to enable> exit
  ```

- **Add throttling support for CoA**: Implements a throttling mechanism within CPC to control the rate (messages per second) at which CoA messages are sent to ASR9K BNG devices, preventing overload. The default state for CoA throttling is enabled.

  - Configure throttling limit:

    ```
    radius advance-tuning throttling-limit <limit>
    ```

  - Enable CoA throttling for ASR9K PEP:

    ```
    radius advance-tuning coa-throttlling-for-asr9k-pep true exit
    ```

  - Disable CoA throttling for ASR9K PEP:

    ```
    radius advance-tuning coa-throttlling-for-asr9k-pep false exit
    ```

- **SRG switchover scenario**: This feature pertains to the handling and behavior during SRG switchover events. The default state is enabled.

  ```
  engine <engine group name> properties com.broadhop.SrgBngSwitchOverEnable value
  <true/false> exit
  ```

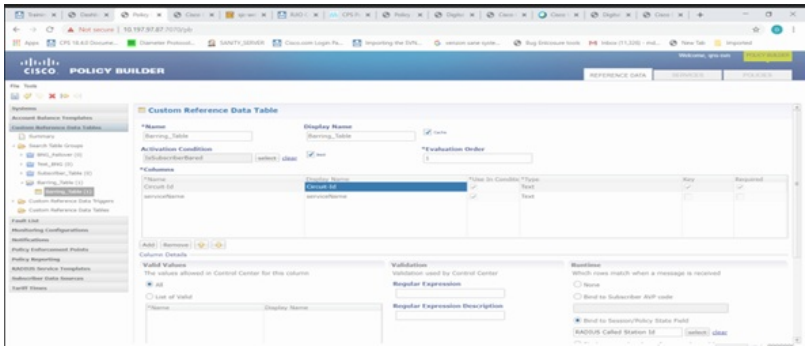
- **PB publish framework modification for audit logging**: Logs every change made during the policy publishing process, whether initiated from the Policy Builder UI or CPS Central UI. This provides a detailed audit trail of policy modifications. The logs are stored in `qns-pb-publish-svn-diff.log` within the `/data/consolidated-aaa-logging/` directory of the `consolidated-aaa-logging-0` pod.

  Ops-Center CLI manages this feature by setting `com.broadhop.pb.publish.audit.feature` to `true` or `false` for the engine group's policy builder. The default state is enabled.

  ```
  engine <engine group name> policy-builder properties com.broadhop.pb.publish.audit.feature
   value <true/false> exit
  ```

- **Call Barring solution**: This feature allows for blocking internet or call access for subscribers connected to specific OTLs during emergency situations. Policy Builder and Control Center's UI manage this feature. It is typically a manual configuration performed on demand during emergency events rather than a persistent on/off toggle.

Figure 23: Call Barrier



- **SOAP call validation**: This feature includes two sub-features to enhance the security and control of SOAP API calls which can be configured in Ops-Center:
  - **White-listed IPs**: Allows to define a list of trusted IPv4 addresses permitted to invoke SOAP calls.

    Example:

    ```
    api unified externalIPs [ 192.0.2.119 192.0.2.120 ]
    ```

  - **Rate Limiting**: Enables to control the number of SOAP API requests made to the Unified API URL within a given time-frame.

    Example:

    ```
    api unified limit-max-requests-per-sec 100
    ```

- **HQoS turbo plan rollout for subscribers**: Facilitates the rollout of HQoS turbo plans to subscribers using an automated Python script (`HQoS_Migration_Script.py`), typically located at `/data/pcf-m13/data-pcf-utilities-0/support/script/`. This script migrates active subscriber sessions in batches, triggers SwitchService and CoA, and logs actions.
  - Enable by executing the `HQoS_Migration_Script.py` script.
  - The feature is disabled if the script is not run.

- **Updating Provisioned-Called-Station-ID based on CSV file**: This feature updates the `Provisioned-Called-Station-Id` for subscribers using the `OLT_rehoming.py` script, typically located at `/data/pcf-m13/data-pcf-utilities-0/support/script/`. The script processes a CSV file containing `networkId` and `calledStationId` pairs to update each subscriber accordingly. Enable it by executing the `OLT_rehoming.py` script. The feature is disabled if the script is not run.
- **CSV report for differences between Called-Station-ID**: This feature generates a Comma Separated Values (CSV) report. It highlights discrepancies between the `Called-Station-ID` in subscriber provisioning data and the active session data (obtained using `cdl show sessions summary`). This helps identify and resolve mismatches. The feature uses the `called_stn_id_report_gen.py` script, typically located at `/data/pcf-m13/data-pcf-utilities-0/support/script/`. Execute the `called_stn_id_report_gen.py` script to enable. The feature is disabled if the script is not run.

Threading configuration for HTTP2 outgoing requests

Configure threading for HTTP2 outgoing requests from cnAAA.

```
advance-tuning async-threading
```

These are the parameters:

Parameter	Description
default-processing-threads processing_threads	Specifies the number of processing threads.The range must be in integers. By default the value is 10.
default-queue-size queue_size	Set the size of the queue. The range of size be in integers. By default the value is 100.
default-worker-threads workerThreads	Define the number of worker. The number must be in integers. By default the value is 20.
max-timeouts-to-reconnect max_timeout_to_reconnect-	Maximum request timeouts to reconnect HTTP2 connections. The timeout must be in integers. By default the value is 0

Note Do not change recommended values.

Sample Configuration:

```
> M7 Performace setup sample  configuration.
[m13-cnaaa/m13] pcf# show running-config advance-tuning async-threading
Wed Jul  16 05:18:44.212 UTC+00:00
advance-tuning async-threading default-worker-threads 25
advance-tuning async-threading default-queue-size 200
advance-tuning async-threading default-processing-threads 12
advance-tuning async-threading http2-connect-timeout-ms 100
advance-tuning async-threading http2-idle-connection-timeout-sec 60
advance-tuning async-threading max-timeouts-to-reconnect 0
[m13-cnaaa/m13] pcf#
```

Istio resource control configuration

Configure istio-resource-control settings for the engine.

```
advance-tuning istio-resource-control engine concurrency count
```

Parameter:

Concurrency Count: Specifies istio-resource-control engine concurrency (Integer, Default: 6).

Note The recommended value is 6, which is the default. This value cannot be changed.

Sample configuration:

```
[m13-cnaaa/m13] pcf# show running-config advance-tuning istio-resource-control
Wed Jul  16 05:25:45.542 UTC+00:00
advance-tuning istio-resource-control engine concurrency 6
[m13-cnaaa/m13] pcf#
```

Redis password configuration

Configure the Redis password for secure access.

```
advance-tuning redis-password password
```

Parameter:

Password: Specifies the Redis password (String).

Usage: Configure the Redis password for secure access.

Sample Configuration:

```
[m13-cnaaa/m13] pcf# show running-config advance-tuning redis-password
Wed Jul  16 06:09:09.895 UTC+00:00
advance-tuning redis-password $8$SjiJVGF1XL25sH4dnOSfs9r9jPKc9paPDISqk3QQubc=
[m13-cnaaa/m13] pcf#
```

Network Slice access control

Enable or disable validation of sliceInfo for PDU sessions.

```
advance-tuning slice-access-control {enable/disable}
```

Parameter:

Enable/Disable: Validates sliceInfo in PDU sessions and rejects unsupported slices (Boolean, Default: false).

OAM Support

This section describes operations, administration, and maintenance information for this feature.

Bulk statistics support

This section provides the list of statistics and counters that are generated for the monitoring for message threshold enhancement. Bulk stat is enabled by default.

Note The following values apply to all the statistics:

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

The following metrics track the counter information:

- inbound_request_threshold_exceeded_total - Captures the total count of the inbound threshold requests exceeded due to overload.

  The following labels are defined for this metric:

  - interface_name
  - service_name
  - operation_name
  - command
  - action

Radius engine CPU configuration

Dynamic resource configuration for engine and radius pods

The cnAAA application supports dynamic resource allocation for engine and radius pods. This feature allows you to tune CPU and memory requests and limits through the Ops-Center CLI, providing the flexibility to adjust resource allocation based on traffic profiles, deployment scale, and environment-specific needs.

This enhancement addresses the limitations of fixed values in deployment templates by providing these benefits:

- Improved performance: You can size resources to match actual traffic demands, which prevents CPU throttling and Out of Memory (OOM) events.
- Enhanced flexibility: The system supports partial configuration, where you can override specific parameters while other parameters remain at default values.
- System stability: The system enforces range validation at the YANG model level to prevent scheduling failures caused by invalid inputs.

Configure CPU and memory resources for engine and radius pods

Use this procedure to dynamically configure CPU and memory resources for engine and radius pods.

Procedure

Step 1 Log in to the Ops-Center CLI and enter configuration mode.

```
config
```

Step 2 Configure resources for engine pods. Replace <profile-name> with the specific engine profile.

```
engine <profile-name> resources requests cpu <cores>
engine <profile-name> resources requests memory <GiB>
```

```
engine <profile-name> resources limits cpu <cores>
engine <profile-name> resources limits memory <GiB>
```

Step 3 Configure resources for radius pods.

```
radius resources requests cpu <cores>
radius resources requests memory <GiB>
radius resources limits cpu <cores>
radius resources limits memory <GiB>
```

Note Perform this configuration when the system is shut down.

Step 4 Commit the changes. The system triggers a rolling rollout of the pods.

```
commit
```

- Engine configuration

```
config
engine production-rjio resources requests cpu 3
engine production-rjio resources requests memory 12
engine production-rjio resources limits cpu 7
engine production-rjio resources limits memory 16
commit
```

- Radius configuration

```
config
radius resources requests cpu 4
radius resources requests memory 16
radius resources limits cpu 8
radius resources limits memory 22
commit
```

Operational considerations

Observe these constraints and behaviors when you configure resource settings:

- Rolling updates: The system applies resource changes through a rolling update strategy. Pods restart sequentially to ensure service continuity. However, it is recommended to perform the configuration change while the system is in a shutdown state.
- Partial configuration: The cnAAA application supports partial configuration. If you configure only one parameter (for example, `cpu limit`), the system retains the existing default values for the remaining parameters.
- Validation ranges: The cnAAA application enforces these input ranges:
  - CPU request: 1 to 5 cores.
  - CPU limit: 5 to 32 cores.
  - Memory request/limit: Represented in GiB.

- Scaling down: While the cnAAA application supports decreasing resource values, we do not recommend scaling down. Reducing resources below current utilization levels may lead to data loss, CPU throttling, or OOM kills.

After you commit changes, verify the rollout by describing the pod status:

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

KPI scheduler

Feature description

Feature Name	Release Information	Description
KPI scheduler	2026.03.0	The KPI scheduler implements a Kubernetes-based leader election mechanism to ensure that only one pod executes the KPI scheduler job in multi-pod deployments. This prevents performance degradation and data inconsistencies caused by redundant scheduler execution. By designating a single leader pod while keeping others in standby, cnAAA optimizes database load and supports automatic failover.

The KPI scheduler implements a leader election mechanism to manage scheduling in multi-pod Kubernetes deployments. Previously, the scheduler executed on every replica pod, which caused redundant database queries, resource waste, and potential data inconsistencies.

This enhancement ensures that only one designated leader pod executes the KPI scheduler job at any given time, while other pods remain in standby. This improves system performance, supports automatic failover, and aligns with Kubernetes best practices for high availability.

Note By default, the leader pod election feature is enabled.

Verify the leader election mechanism

Use this procedure to validate that the leader election mechanism.

Procedure

Step 1 Verify that all the engine replica pods are up and running. Use the `kubectl get pods -n <namespace>` command.

Step 2 You can identify the active engine leader pod in two ways:

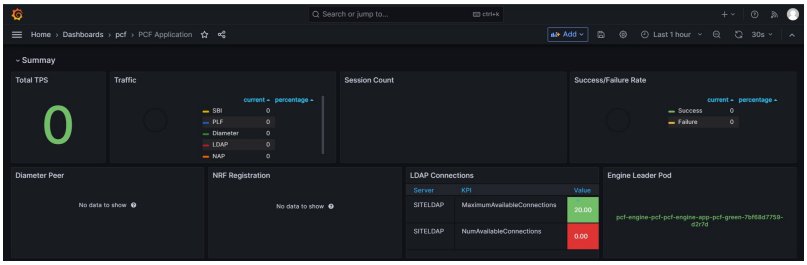
- Executing `show engine_leader` in the Ops-Center.

Example:

```
[unknown] pcf#
[unknown] pcf# show engine_leader
Thu Jul  9  11:30:50.610 UTC+00:00
PODNAME
------------------------------------------------------------
pcf-engine-pcf-pcf-engine-app-pcf-green-7bf68d7759-d2r7d

[unknown] pcf#
```

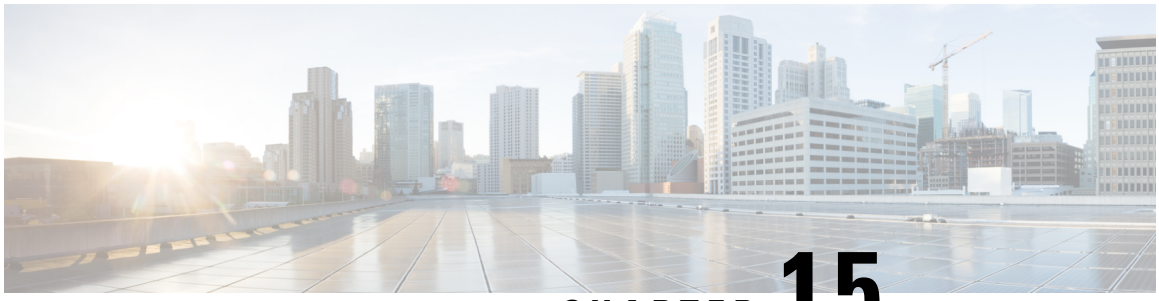
- Verify in Grafana dashboard.



Step 3 Confirm that the KPI scheduler job executes at the 5-minute interval by monitoring the leader pod logs with this command.

```
kubectl logs -f <leader-pod-name> -n <namespace> | grep "PersistentKPIScheduler"
```

Note If a leader pod is executing the KPI scheduler and is deleted or crashes due to an issue, another replica pod becomes the leader.



CHAPTER 15

cnAAA application based alerts

- Feature Summary and Revision History, on page 259
- Feature Description, on page 260
- How it Works, on page 260
- Configure alert rules, on page 260
- Sample alerts configuration, on page 263
- Support notifications for system and application alarms, on page 268

Feature Summary and Revision History

Summary Data

Table 36: Summary Data

Applicable Products or Functional Area	cnAAA
Applicable Platform(s)	SMI
Feature Default Setting	Disabled – Configuration required to enable
Related Documentation	Not Applicable

Feature History

Table 37: Feature History

Feature Details	Release
First introduced.	2025.01.0

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 the following 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.

View alert logger

The alert logger stores the alerts that cnAAA 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.

This section provides sample outputs for active and historical alerts in the cnAAA system.

- **Active Alerts Summary**

  To view a summary of active alerts, use this command:

  ```
  [cpc-cluster] cee# show alerts active summary
  ```

  Sample output:

  ```
  Mon Oct  27 12:11:20.261 UTC+00:00
  alerts active summary db-no-tx dc6b09b14d2e
   severity minor
   startsAt 10-27T12:09:33
   source   System
   summary  Unknown
  alerts active summary k8s-pod-restarting 7849e8c4a6e8
   severity minor
  ```

```
 startsAt 10-27T12:07:21
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-product..."
alerts active summary k8s-pod-crashing-loop 0007305db5f6
 severity critical
 startsAt 10-27T12:07:11
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-product..."
alerts active summary k8s-pod-crashing-loop ff914a2e3df1
 severity critical
 startsAt 10-27T12:06:51
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/radius-ep-5 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop 7ddadee4531f
 severity critical
 startsAt 10-27T12:06:51
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/radius-ep-2 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop a910267b6f7a
 severity critical
 startsAt 10-27T12:06:51
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/radius-ep-0 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop 7ff2a7486644
 severity critical
startsAt 10-27T12:06:51
 source   cpc-cluster-master-1
 summary  "Pod cpc-m13/radius-ep-1 (radius-ep) is restarting ..."
```

- **Active Alerts Detail**

  To view detailed information for active alerts, use this command:

  ```
  [cpc-cluster] cee# show alerts active detail
  ```

  Sample output:

  ```
  Mon Oct  27 12:11:09.961 UTC+00:00
  alerts active detail db-no-tx dc6b09b14d2e
   severity    minor
   type        "Processing Error Alarm"
   startsAt    2025-10-27T12:09:33.265Z
   source      System
   summary     Unknown
   labels     [ "alertname: db-no-tx" "cluster: cpc-cluster_cee-m13" "monitor: prometheus"
   "replica: cpc-cluster_cee-m13" "severity: minor" ]
   annotations [ "type: Processing Error Alarm" ]
  alerts active detail k8s-pod-restarting 7849e8c4a6e8
  severity    minor
   type        "Processing Error Alarm"
   startsAt    2025-10-27T12:07:21.119Z
   source      cpc-cluster-master-1
   summary    "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp
   (crd) is restarting 4.03 times / 10 minutes."
   labels      [ "alertname: k8s-pod-restarting" "chartName: metrics" "cluster:
  cpc-cluster_cee-m13" "component: kube-state-metrics" "container: crd" "hostname:
  cpc-cluster-master-1" "instance: 192.102.0.105:8080" "job: kubernetes-pods" "monitor:
  prometheus" "namespace: cpc-m13" "pod:
  crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp" "pod_template_hash:
  755686975c" "release: cee-m13-cnat-monitoring" "replica: cpc-cluster_cee-m13" "severity:
   minor" "uid: 56c9e482-b47c-40ea-bc97-49698f8b74ee" ]
  annotations [ "summary: Pod
  cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp (crd) is
  restarting 4.03 times / 10 minutes." "type: Processing Error Alarm" ]
  ```

- **Historical Alerts Count**

  To view a count of historical alerts by severity, use this command:

  ```
  [cpc-cluster] cee# show alerts history count
  ```

  Sample output:

  ```
  Mon Oct 27 12:44:36.463 UTC+00:00
  SEVERITY TOTAL

  minor 152
  major 130
  critical 532
  ```

- **Historical Alerts Detail Summary**

  To view a summary of historical alerts in detail, use this command:

  ```
  [cpc-cluster] cee# show alerts history detail summary
  ```

  Sample output:

  ```
  Mon Oct 27 12:45:05.478 UTC+00:00
  alerts history detail db-no-tx dc6b09b14d2e
  summary Unknown
  alerts history detail k8s-pod-restarting 7849e8c4a6e8
  summary "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp
  (crd) is restarting 1.02 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop 0007305db5f6
  summary "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp
  (crd) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop 7ff2a7486644
  summary "Pod cpc-m13/radius-ep-4 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop 7ddadee4531f
  summary "Pod cpc-m13/radius-ep-2 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop ff914a2e3df1
  summary "Pod cpc-m13/radius-ep-5 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop 1d24967fc160
  summary "Pod cpc-m13/radius-ep-1 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop 162420e4043a
  summary "Pod cpc-m13/radius-ep-3 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-crashing-loop a910267b6f7a
  summary "Pod cpc-m13/radius-ep-0 (radius-ep) is restarting 2.03 times / 10 minutes."
  alerts history detail k8s-pod-restarting b9840e7118da
  summary "Pod cpc-m13/radius-ep-4 (radius-ep) is restarting 1.02 times / 10 minutes."
  ```

Sample alerts configuration

This section provides sample configurations that are defined in cnAAA.

Process-Level alerts

CDL Endpoint down

Use the following 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 the following 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
```

ETCD State Change

Use the following commands to configure alerts related to etcd state change.

```
alerts rules group ep-mapping-change
   rule pod-down
   expression up{pod=~'etcd-cnAAA.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value EP Mapping Change
   exit
exit
```

Grafana Dashboard State Change

Use the following 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 the following 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
```

cnAAA Engine State Change

Use the following commands to configure alerts related to cnAAA Engine state change.

```
alerts rules group cnAAA-engine-change
   rule pod-down
   expression up{pod=~'cnAAA-engine-cnAAA.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value cnAAA Engine Changed
   exit
exit
```

RADIUS Endpoint State Change

Use the following commands to configure alerts related to RADIUS endpoint state change.

```
alerts rules group cnAAA-RADIUS-ep-change
   rule pod-down
   expression up{pod=~'cnAAA-RADIUS-ep.*'} == 0
   duration 1m
   severity major
   type Equipment Alarm
   annotation description
   value cnAAA RADIUS EP Change
   exit
exit
```

Call Flow procedure alerts

PLF query request

Use the following commands to configure alerts related to PLF Query Request.

```
alerts rules group cnAAAProcStatus
   interval-seconds 300
```

```
   rule PLFRequest
   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 the following commands to configure alerts related to NAP Notification Request.

```
alerts rules group cnAAAProcStatus
   interval-seconds 300
   rule NAPNotification
   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
```

Support notifications for system and application alarms

Revision History

Feature Name	Release Information	Description
Notification and Alarms Support	2025.02	This feature monitors policy application during startup and operations to ensure system stability. It triggers alarms on configuration failures, initiating actions to maintain system reliability through notifications.
Support Notifications for System and Application Alarms	2025.01	The Monitoring and Alert Notification framework enhances system and application reliability by issuing SNMP traps to alert critical events. It categorizes alerts into proactive (requiring attention) and reactive (indicating past events), covering cnAAA issues, ensuring timely interventions for network stability.

Overview

The Monitoring and Alert Notification framework enhances system and application reliability by tracking events and sending alerts. It notifies network administrators of critical events and changes, enabling timely interventions to maintain system stability. The framework provides SNMP notification traps, categorized into proactive and reactive alerts. Proactive traps are alerts based on system events or changes that require attention, while reactive traps notify administrators of events that have already occurred. Alerts cover issues in cnAAA, database problems, load averages, and link status.

The Notification and Alarms ensures policy configurations are correctly applied during startup and ongoing operation. It identifies configuration failures and raises critical alarms to maintain system stability. This feature monitors and notifies two key alarms: "PoliciesNotConfigured" and "PolicyConfiguration."

System Alarms

System alarm configurations are required to configure in cee-ops-center within cnAAA deployment, following are the examples of alarms for DB, CPU etc.

- **All-SCDB-DB-Members-Down:**

  Description: Connectivity failure to all members of the SCDB, ADMIN, and SPR replica sets.

  Expression:sum(mongo_node_state{replica_set=~'sdb.*'}) == 0

- **All-ADMIN-DB-Members-Down:**

Description: This alert is triggered when all DB members of admin are down.

Expression: sum(mongo_node_state{replica_set=~'admin.*'}) == 0

- **All-SPR-DB-Members-Down:**

  Description: This alert is triggered when all DB members of SPR are down.

  Expression:sum(mongo_node_state{replica_set=~'spr.*'}) == 0

- **Primary-SCDB-Member-Down:**

  Description: Inability to locate the primary member for the SCDB sets.

  Expression:mongo_primary_reachable{replica_set=~'sdb.*'} == 0

- **Primary-SPR-DB-Member-Down:**

  Description: Inability to locate the primary member for the SPR replica sets.

  Expression: mongo_primary_reachable{replica_set=~'spr.*'} == 0

- **Primary-Admin-DB-Member-Down:**

  Description: Inability to locate the primary member for the admin replica sets.

  Expression: mongo_primary_reachable{replica_set=~'admin.*'} == 0

- **Secondary-Admin-DB-Member-Down:**

  Description: Inability to locate the secondary member for the admin sets.

  Expression: sum(mongo_node_state{replica_set=~'admin.*', state='secondary'}) == 0

- **Secondary-SCDB-Member-Down:**

  Description: Inability to locate the secondary member for the SCDB sets.

  Expression: sum(mongo_node_state{replica_set=~'sdb.*', state='secondary'}) == 0

- **Secondary-SPR-DB-Member-Down:**

  Description: Inability to locate the secondary member for the SPR replica sets.

  Expression: sum(mongo_node_state{replica_set=~'spr.*', state='secondary'}) == 0

- **load-average-high:**

  Description: This alarm is triggered when the the system's load average surpasses a configured threshold for over 5 minutes.

  Expression: node_load5 > 5

- **Link-Down:**

  Description: This alarm is triggered when the connectivity or ping failure to a system-attached physical interface.

  Expression: node_network_up{device = 'ens192'} == 0

- **Replication-lag:**

  Description: This alarm is triggered when the replication lag exceeds 2 seconds.

  Expression: mongo_replication_lag_seconds > 2

The Alert Logger records all generated Alarm by default, accessible through specific show commands for reviewing stored Alarm:

- show alert history {detail | summary}
- show alert active {detail | summary}

Monitoring alarm support for SVN repository

This feature monitors critical Subversion (SVN) repositories, detecting if any essential repository or `.broadhopFileRepository` file with in the essential repository is removed. As an early warning system, it prevents potential system outages and ensures development environment integrity. When a deletion occurs, the system triggers a critical alarm, which helps in minimize service disruption.

Configure critical SVN repository monitoring in cnAAA Ops-Center

Procedure

Follow these steps to configure critical SVN repository monitoring.

Step 1 Log in to cnAAA Ops-Center and enter the configuration mode.

```
config
```

Step 2 Enter this show command to display all the configured engine properties and their values.

```
show full-configuration engine <engine-group> properties
```

Step 3 Define the list of SVN repositories to monitor.

```
properties list.of.repos.to.be.monitored
value <repository-names>
exit
```

Note Each repository name must be separated by a comma (,) with no spaces between names.

Step 4 Set the frequency for the system checks of the SVN repositories.

```
properties com.broadhop.svn.monitoring.interval.ms
value <interval-in-ms>
exit
```

Note The default interval is 300,000 milliseconds (five minutes).

Example

```
config
# Define the list of SVN repositories to monitor.
engine cpc-green properties list.of.repos.to.be.monitored
value new_import8,new_import9,new_import10,new_import11
exit

# Set the frequency for the system checks of the SVN repositories.
engine cpc-green properties com.broadhop.svn.monitoring.interval.ms
value 40000
exit
```

Configure alarm for monitoring critical SVN repository

Procedure

Follow these steps to configure critical SVN repository monitoring.

Step 1 Login to CEE Ops-Center and enter the configuration mode.

```
config
```

Step 2 Enter the show command to display all available alerts.

```
Show full configuration alerts
```

Step 3 Create two alarm rules to trigger alerts when a repository or the `.broadhopFileRepository` file is deleted.

a) Alarm for SVN repository deletion: This alarm triggers when a monitored SVN repository is completely removed.

```
alerts rules group svn-repo-deleted
rule SvnRepoDeleted
expression "sum(svn_repo_deleted_total{}!= 0) by (message_type) "
severity critical
type "Communications Alarm"
annotation summary
value "Expected SVN Repository missing is: {{ $labels.message_type }} "
exit
exit
exit
```

Sample Alarm Output:

```
 alerts active detail SvnRepoDeleted 8802f40d4e65
 severity    critical
 type        "Communications Alarm"
 startsAt    2025-10-27T09:12:32.190Z
 source      System
 summary     " Expected SVN Repository missing is: TestPB1 "
 labels     [ "alertname: SvnRepoDeleted" "cluster: unknown_cee" "message_type: TestPB1" "monitor:
 prometheus" "replica: unknown_cee" "severity: critical" ]
 annotations [ "summary:  Expected SVN Repository missing is: TestPB1 " "type: Communications
Alarm" ]
```

b) Alarm for `.broadhopFileRepository` file deletion: This alarm triggers when the .broadhopFileRepository file is missing from a monitored SVN repository.

```
alerts rules group config-file-deleted-from-svn-repo
rule ConfigFileDeletedFromSvnRepo
```

```
expression "sum(config_file_deleted_from_svn_repo_total{}!=0) by (message_type) "
severity critical
type "Communications Alarm"
annotation summary
value "Expected SVN Repository missing the metadata is: {{ $labels.message_type }} "
exit
exit
exit
```

Sample Alarm Output:

```
 alerts active detail ConfigFileDeletedFromSvnRepo 90a6cec6b372
 severity    critical
 type        "Communications Alarm"
 startsAt    2025-10-27T09:40:23.610Z
 source      System
 summary     " Expected SVN Repository missing  the metadata is: TestPB "
 labels      [ "alertname: ConfigFileDeletedFromSvnRepo" "cluster: unknown_cee" "message_type:
TestPB" "monitor: prometheus" "replica: unknown_cee" "severity: critical" ]
 annotations [ "summary:  Expected SVN Repository missing  the metadata is: TestPB " "type:
Communications Alarm" ]
```

Application alarms

Application alarm configurations are required to configure in cee-ops-center within cnAAA deployment, following are the examples of application Alarm:

- **Access Reject**:

  Description: This alarm is triggered when the number of Access Reject messages exceeds a threshold limit.

  Expression:sum(rate(radius_responses_total{message_type="AccessReject"}[1m])) > N

- **Service Stop Request**:

  Description: This alarm is triggered when the number of Service Stop Requests exceeds a threshold limit.

  Expression:avg(rate(radius_accounting_request_total{accountingType="SessionAccounting", statusType="Stop"}[1m])) > N

- **Service Stop Response**:

  Description: This alarm is triggered when the number of Service Stop Responses exceeds a threshold limit.

  Expression:sum(rate(radius_accounting_response_total{accountingType="SessionAccounting", statusType="Stop"}[1m])) > N

- **PER EP TPS Radius**:

  Description: This alarm is triggered when the Transactions Per Second (TPS) exceeds a threshold limit.

  Expression:sum(irate(radius_requests_total[1m]) or vector(0)) + sum(irate(radius_accounting_request_total[1m]) or vector(0)) by (pod)

- **Memory Used in Radius POD**:

  Description: This alarm is triggered when the memory usage in a Radius POD exceeds a threshold limit.

Expression:sum(rate(jvm_memory_bytes_used{namespace="pcf", component="cps-radius-ep"}[1m]))

- **GC Time Period**:

  Description: This alarm is triggered when the garbage collection (GC) time exceeds a threshold limit.

  Expression:

  increase(jvm_gc_collection_seconds_sum{component="cps-radius-ep",namespace="pcf"}[$__interval])
- **Total Radius Auth Messages Overload Rejected**:

  Description: This alarm is triggered when the number of Radius auth messages rejected due to overload exceeds a threshold limit.

  Expression:

  sum(rate(total_radius_auth_messages_overload_rejected{message_type="AccessReject"}[1m])) > N
- **Total Radius Messages Overload Dropped on Session Accounting**

  Description: This alarm is triggered when the number of Radius messages dropped on Session Accounting due to overload exceeds a threshold limit.

  Expression:

  sum(rate(total_radius_messages_overload_dropped{message_type="SessionAccounting",status_type="Start"}[1m])) > N
- **Total Radius Messages Overload Dropped on Service Accounting**

  Description: This alarm is triggered when the number of Radius messages dropped on Service Accounting due to overload exceeds a threshold limit.

  Expression:

  sum(rate(total_radius_messages_overload_dropped{message_type="ServiceAccounting",status_type="Start"}[1m])) > N
- **GRPC Message Send Total on Accounting Request**

  Description: This alarm is triggered when the GRPC message send total for Accounting Request exceeds a threshold limit.

  Expression:

  sum(rate(grpc_message_send_total{message_type="AccountingRequest"}[1m])) > N
- **GRPC Message Send Total on Access Request**

  Description: This alarm is triggered when the GRPC message send total for Access Request exceeds a threshold limit.

  Expression: sum(rate(grpc_message_send_total{message_type="AccessRequest"}[1m])) > N

- **Radius Proxy Accounting Response Total on Error**

  Description: This alarm is triggered when the Radius proxy accounting responses with errors exceed a threshold limit.

  Expression:
  sum(rate(radius_proxy_accounting_response_total{accounting_type="ServiceAccounting",status_type="Start",result="ERROR"}[1m])) > 1

- **CoA Timeout**

Description: This alarm is triggered when Change of Authorization (CoA) timeouts exceed a threshold limit.

Expression: sum(rate(radius_request_timeout_total{message_type="CoaRequest"}[1m])) > N

Policy Engine Timeout Message on Accounting Request

Description: This alarm is triggered when Policy Engine timeout messages for Accounting Request exceed a threshold limit.

Expression:
sum(rate(POLICY_ENGINE_TIMEOUT_MESSAGE{message_type="AccountingRequest"}[1m])) > N

- **Policy Engine Timeout Message on Access Request**

  Description: This alarm is triggered when Policy Engine timeout messages for Access Request exceed a threshold limit.

  Expression:
  sum(rate(POLICY_ENGINE_TIMEOUT_MESSAGE{message_type="AccessRequest"}[1m])) > N

  Policy Engine Message Total on Access Request

  Description: This alarm is triggered when the total Policy Engine messages for Access Request exceed a threshold limit.

  Expression: sum(rate(policy_engine_message_total{message_type="AccessRequest"}[1m])) > N

  Policy Engine Message Total on Accounting Request

  Description: This alarm is triggered when the total Policy Engine messages for Accounting Request exceed a threshold limit.

  Expression: sum(rate(policy_engine_message_total{message_type="AccountingRequest"}[1m])) > N

- **Dispatch Error Total on Bundled CoA Request**

  Description: This alarm is triggered when dispatch errors for Bundled Change of Authorization (CoA) Requests exceed a threshold limit.

  Expression: sum(rate(dispatch_error_total{message_type="AsyncCoARequest"}[1m])) > N

- **Dispatch Error Total on AsyncCoA Request**

  Description: This alarm is triggered when dispatch errors for Async Change of Authorization (CoA) Requests exceed a threshold limit.

  Expression: sum(rate(dispatch_error_total{message_type="BundledCoARequest"}[1m])) > N

- **Process Message Total on Accounting Response**

  Description: This alarm is triggered when the total processed messages for Accounting Response exceed a threshold limit.

  Expression: sum(rate(process_message_total{message_type="AccountingResponse"}[1m])) > N

- **Process Message Total on Access Accept**

  Description: This alarm is triggered when the total processed messages for Access Accept exceed a threshold limit.

  Expression: sum(rate(process_message_total{message_type="AccessAccept"}[1m])) > N

- **Outbound Request Total on Proxy Accounting**

  Description: This alarm is triggered when the total outbound requests for Proxy Accounting exceed a threshold limit.

  Expression: sum(rate(outbound_request_total{message_type="ProxyAccounting"}[1m])) > N

- **Outbound Request Total on CoA Request**

  Description: This alarm is triggered when the total outbound requests for Change of Authorization (CoA) Requests exceed a threshold limit.

  Expression: sum(rate(outbound_request_total{message_type="CoARequest"}[1m])) > N

- **Inbound Request Total on Proxy Accounting**

  Description: This alarm is triggered when the total inbound requests for Proxy Accounting exceed a threshold limit.

  Expression: sum(rate(inbound_request_total{message_type="ProxyAccounting"}[1m])) > N

- **Inbound Request Total on Access Request**

  Description: This alarm is triggered when the total inbound requests for Access Request exceed a threshold limit.

  Expression:sum(rate(inbound_request_total{message_type="AccessRequest"}[1m])) > N

- **Record Conflict Merge Total**

  Description: This alarm is triggered when the total record conflict merges exceed a threshold limit.

  Expression:sum(rate(record_conflict_merge_total[1m])) > N

- **Radius Access Request Message on Error**

  Description: This alarm is triggered when Radius Access Request messages on errors exceed a threshold limit.

  Expression: sum(rate(message_total{type="radius-access-request-message", status="error"}[1m])) > N

- **I Send Access Accept on Error**

  Description: This alarm is triggered when "I Send Access Accept" messages on errors exceed a threshold limit.

  Expression:sum(rate(action_total{type="i-send-access-accept", status="error"}[1m]))

- **Radius Accounting Message on Error**

  Description: This alarm is triggered when Radius Accounting messages on errors exceed a threshold limit.

  Expression:sum(rate(message_total{type="radius-accounting-message", status="error"}[1m])) > N

Critical Alert to be Raised RADIUS Pod not Sending Access-Accept

This feature monitors RADIUS pods and alerts when they stop sending **access-accept** responses. It uses Prometheus to collect metrics and an alert rule to trigger a critical alert if a pod fails to respond for a set time. This ensures rapid issue detection, resolution, and improved RADIUS service reliability.

How RADIUS pod access-accept monitoring works

This feature operates through an integrated monitoring pipeline:

1. **Metric export**: Each RADIUS pod exports operational metrics, including `radius_responses_total`. This metric tracks the cumulative count of RADIUS responses, with `message_type="AccessAccept"` indicating successful responses..
2. **Prometheus data collection**: A Prometheus instance, deployed in the CEE namespace, collects these metrics from all RADIUS pods, typically in the CPC namespace. Prometheus continuously collects this time-series data.
3. **Real-time rate calculation**: Prometheus calculates the rate of access-accept responses over a one-minute window for each RADIUS pod. It uses the `rate()` function on the `radius_responses_total{message_type="AccessAccept"}` metric.
4. **Alert condition evaluation**: A predefined Prometheus alert rule evaluates this calculated rate. If the rate of access-accept responses for any pod drops to zero and remains at zero for one continuous minute, the alert condition is met.
5. **Critical alert trigger**: If the alert condition is met, the system triggers a critical alert named `RadiusPodNotSendingAccessAccept`. This alert includes detailed annotations identifying the specific pod and its namespace that is experiencing the issue.
6. **Automatic alert resolution**: If the affected RADIUS pod resumes sending access-accept responses, the rate increases above zero. The Prometheus alert rule automatically resolves the critical alert.
7. **Visualization (Optional)**: All collected metrics and alert states can be visualized on a Grafana dashboard. This provides real-time insights into RADIUS service health and performance.

RADIUS Pod access-accept alert

RADIUS pods export metrics, specifically `radius_responses_total` with the `message_type="AccessAccept"` label. This metric is crucial for data collection.

The cnAAA system is configured to raise a critical alert when a RADIUS pod does not send access-accept responses:

```
groups:
- name: RadiusAlerts
  rules:
  - alert: RadiusPodNotSendingAccessAccept
    expr: sum(rate(radius_responses_total{message_type="AccessAccept"}[1m])) by (pod,
namespace) == 0
    for: 1m # The condition must persist for one minute before the alert is triggered.
    labels:
      severity: critical
    annotations:
      summary: "Radius pod not sending AccessAccept"
      description: "The pod {{ $labels.pod }} in namespace {{ $labels.namespace }} is not
sending AccessAccept responses."
```

Configure Grafana dashboard for RADIUS access-accept:

- Ensure Grafana is configured with Prometheus as a data source.
- Create new panels on a Grafana dashboard to visualize the `radius_responses_total` metric.

  Query for Access-Accept rate per pod:

```
sum(rate(radius_responses_total{message_type="AccessAccept"}[1m])) by (pod, namespace)
 == 0
```

This section provides an example of the `RadiusPodNotSendingAccessAccept` alert.

To view the alert history, use this command:

```
cee# show alerts history summary | tab | include Radius

RadiusPodNotSendingAc  7eb5eacfa11f  critical  10-23T05:12:58  15m20s    radius-ep-1
      Radius pod not sending AccessAccept
RadiusPodNotSendingAc  16e12fc1f5c0  critical  10-23T05:04:18  2m50s     radius-ep-2
      Radius pod not sending AccessAccept
RadiusPodNotSendingAc  fbcc7250ecbf  critical  10-23T04:51:48  2m20s     radius-ep-1
      Radius pod not sending AccessAccept
```

The alert clears once the RADIUS endpoint pod resumes sending Access-Accept responses.

To view active alerts, use this command:

```
cee# show alerts active summary | tab | include Radius
```

Note

- If no traffic runs, alerts are present by default for all RADIUS pods. Alerts clear when traffic resumes.
- This alert also generates when no RADIUS endpoint pod entries appear in ETCD. This indicates that no RADIUS endpoint pods handle traffic.

Follow these steps to troubleshoot `RadiusPodNotSendingAccessAccept` alerts.

- Inspect the RADIUS pod and its logs. Describe the RADIUS pod and check its logs for error-related information.

  ```
  kubectl describe pod <radius-pod-name> -n <namespace>
  kubectl logs <radius-pod-name> -n <namespace>
  ```

- Restart or delete the RADIUS pod. If issues persist with the RADIUS pod, restart or delete it.

  ```
  kubectl rollout restart deployment/<radius-deployment-name> -n <namespace>
  # OR
  kubectl delete pod <radius-pod-name> -n <namespace>
  ```

Configuration

Alert rules can be configured using the following commands:

```
configure
    alerts rules group alert_group_name
    interval-seconds seconds
    rule rule_name
        expression promql_expression
        duration duration
        severity severity_level
        type alert_type
        annotation annotation_name
        value annotation_value
    exit
exit
```

Key Configuration Parameters

- alert_group_name: Name of the alert group, up to 64 characters.
- interval-seconds: The evaluation interval in seconds.
- rule_name: Name of the alerting rule.
- promql_expression: PromQL syntax-based alert rule query.
- duration: Duration before an alert condition is considered true.
- severity_level: Urgency level, ranging from critical to warning.
- alert_type: User-defined alert types, e.g., Operational Violation, Security Service.
- annotation_name/value: Annotations attached to alerts.

Policy configuration counters

Policy configurations during system startup and policy structure changes are monitored using engine counters to ensure system stability.

To view the counters, use this command:

```
kubectl exec -it <engine-pod> -n <namespace> -- curl -G http://127.0.0.1:8080/metrics |
grep poli
```

- **Set up policies_not_configured_total Counter**

  This counter tracks the success or failure of policy configurations during system startup. It increments when policy configuration fails and resets when successful.

  Example:

  ```
  # HELP policies_not_configured_total Total of policies_not_configured
  # TYPE policies_not_configured_total counter
  policies_not_configured_total{node_type="unknown", message_type="Policies Not
  Configured",}
  ```

- **last_policy_configuration_failed_total**

  This counter monitors changes to the system policy structure. It increments if the last policy configuration attempt fails and resets upon success.

  Example:

  ```
  # HELP last_policy_configuration_failed_total Total of last_policy_configuration_failed
  # TYPE last_policy_configuration_failed_total counter
  last_policy_configuration_failed_total{node_type="unknown", message_type="Last Policy
  Configuration Failed due to java.lang.ArithmeticException / by zero",}
  ```

Alarms

This section details alarms triggered for policy configuration issues and highlights the need to monitor and resolve these alerts to maintain cnAAA system functionality.

- **Policies not configured**

  This alarm activates when the policy engine cannot locate any applicable policies during startup. It is critical and requires immediate attention to ensure system services function correctly.

Formula:

```
sum(policies_not_configured_total) != 0
```

- **Policy configuration**

  This alarm activates when a change to the system policy structure fails. Although the system remains stable and operational, it is advisable to check the notification's additional information to determine if further investigation is needed.

  Formula:

  ```
  sum(last_policy_configuration_failed_total{}!=0) by (message_type)
  ```

CEE configuration:

```
alerts rules group policy-config
  rule PoliciesNotConfigured
    expression "sum(policies_not_configured_total) != 0"
    severity   critical
    type       "Communications alarm"
    annotation summary
    value "Policies not configured"
    exit
  exit
exit

alerts rules group last-policy-config
  rule PolicyConfiguration
     expression "sum(last_policy_configuration_failed_total{}!=0) by (message_type)"
     severity   critical
     type       "Communications alarm"
     annotation summary
     value "{{ $labels.message_type }}"
     exit
  exit
exit
```

Check active alerts

Follow these steps to check active alerts:

Procedure

Step 1 Enter the following command in the CLI

```
show alerts active ?
```

Sample alert output:

This section provides sample outputs for active alerts in the cnAAA system.

- **Active Alerts Count**

  ```
  [cpc-cluster] cee# show alerts active count
  Sample output:
  ```
  ```
  ```

```
Mon Oct 27 12:11:31.463 UTC+00:00
SEVERITY TOTAL

minor 12
major 4
critical 15
```

- **Active Alerts Summary**

```
[cpc-cluster] cee# show alerts active summary
Mon Oct 27 12:11:20.261 UTC+00:00
alerts active summary db-no-tx dc6b09b14d2e
severity minor
startsAt 10-27T12:09:33
source System
summary Unknown
alerts active summary k8s-pod-restarting 7849e8c4a6e8
severity minor
startsAt 10-27T12:07:21
source cpc-cluster-master-1
summary "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-product..."
alerts active summary k8s-pod-crashing-loop 0007305db5f6
severity critical
startsAt 10-27T12:07:11
source cpc-cluster-master-1
summary "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-product..."
alerts active summary k8s-pod-crashing-loop ff914a2e3df1
severity critical
startsAt 10-27T12:06:51
source cpc-cluster-master-1
summary "Pod cpc-m13/radius-ep-5 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop 7ddadee4531f
severity critical
startsAt 10-27T12:06:51
source cpc-cluster-master-1
summary "Pod cpc-m13/radius-ep-2 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop a910267b6f7a
severity critical
startsAt 10-27T12:06:51
source cpc-cluster-master-1
summary "Pod cpc-m13/radius-ep-0 (radius-ep) is restarting ..."
alerts active summary k8s-pod-crashing-loop 7ff2a7486644
severity critical
startsAt 10-27T12:06:51
source cpc-cluster-master-1
summary "Pod cpc-m13/radius-ep-1 (radius-ep) is restarting ..."
```

- **Active Alerts Detail**

```
[cpc-cluster] cee# show alerts active detail
Mon Oct 27 12:11:09.961 UTC+00:00
alerts active detail db-no-tx dc6b09b14d2e
severity minor
type "Processing Error Alarm"
startsAt 2025-10-27T12:09:33.265Z
source System
summary Unknown
labels [ "alertname: db-no-tx" "cluster: cpc-cluster_cee-m13" "monitor: prometheus" "replica:
cpc-cluster_cee-m13" "severity: minor" ]
annotations [ "type: Processing Error Alarm" ]
alerts active detail k8s-pod-restarting 7849e8c4a6e8
severity minor
type "Processing Error Alarm"
startsAt 2025-10-27T12:07:21.119Z
source cpc-cluster-master-1
```

```
summary "Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp (crd) is
restarting 4.03 times / 10 minutes."
labels [ "alertname: k8s-pod-restarting" "chartName: metrics" "cluster: cpc-cluster_cee-m13"
"component: kube-state-metrics" "container: crd" "hostname: cpc-cluster-master-1" "instance:
192.102.0.105:8080" "job: kubernetes-pods" "monitor: prometheus" "namespace: cpc-m13" "pod:
crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp" "pod_template_hash: 755686975c"
 "release: cee-m13-cnat-monitoring" "replica: cpc-cluster_cee-m13" "severity: minor" "uid:
56c9e482-b47c-40ea-bc97-49698f8b74ee" ]
annotations [ "summary: Pod cpc-m13/crd-api-cpc-m13-cpc-engine-app-production-rjio-7689c94786-774gp
 (crd) is restarting 4.03 times / 10 minutes." "type: Processing Error Alarm" ]
```

Step 2 Choose one of the following options to view the alerts:

```
count     Count Version
detail    Detailed Version
summary   Compact Version
```

Note For more information on ULB, see the *CEE Configuration and Administration Guide*.

Add Hostname in SNMP Traps

The Monitoring and Alert Notification framework improves system reliability by tracking events and sending alerts to network administrators. It enables timely responses to critical events and system changes to maintain stability. The framework uses Simple Network Management Protocol (SNMP) notification traps, which include:

- Proactive traps: Alerts based on system events or changes that require attention.
- Reactive traps: Alerts for events that have already occurred.

Configure SNMP trap alerts with hostname

To enhance monitoring by including the hostname in SNMP traps and ensure NMS identify the device that sent each trap, follow these configuration and verification steps.

Procedure

Step 1 Log in to the CEE Ops Center.

Step 2 Configure basic SNMP trapper settings.

- Enable the SNMP trapper:

  ```
  snmp-trapper enable true
  ```

- Define the SNMP version 2c (v2c) target receivers for both IPv4 and IPv6 by specifying the NMS IP address, port , and community string:

  Sample configuration for an IPv4 target:

```
snmp-trapper v2c-target 10.1.36.96
 port 162
 community Re4D0nLy5TrinG
exit
```

Sample configuration for an IPv6 target:

```
snmp-trapper v2c-target 1111::10:1:47:9
 port 162
 community Re4D0nLy5TrinG
exit
```

Step 3 Create specific alert rules to trigger SNMP traps based on conditions such as CPU utilization.

Sample configuration:

```
alerts rules group cpu-util-high
 rule cpu-util-idle
  expression "avg(rate(node_cpu_seconds_total{mode='idle'}[1m])) by (hostname) *100 > 10"
  duration 20s
  severity critical
  type "Processing Error Alarm"
  annotation description
   value "High CPU"
  exit
 exit
exit
```

Note These rules determine which events will generate alerts for SNMP traps.

Step 4 Specify the source IP addresses from which SNMP traps should originate.

Sample configuration:

```
snmp-trapper source-ip-routes internal-vip 10.192.2.31
snmp-trapper source-ip-routes default-external-vip 10.84.117.99
snmp-trapper source-ip-routes source-external-vips cee-gamma-cncps-cee
 external-vip 10.84.117.99
exit
```

Note This ensures SNMP traps are sent from the correct interfaces or Virtual IPs.

Step 5 Enter the `TCP dump` command to verify the alerts.

Sample configuration:

```
2025-05-21 11:47:12.330521 IP 10.1.46.84.49592 > 10.1.36.96.162:  V2Trap(706)
 .1.3.6.1.2.1.1.3.0=321056 .1.3.6.1.6.3.1.1.4.1.0=.1.3.6.1.4.1.9.9.999.0.1
.1.3.6.1.4.1.9.9.999.1.1="cpu-util-idle"
.1.3.6.1.4.1.9.9.999.1.2="/rid7802257-brprakas-1-master3" .1.3.6.1.4.1.9.9.999.1.3="critical"
 .1.3.6.1.4.1.9.9.999.1.4=07_e9_05_15_0b_2d_16_00_2b_00_00 .1.3.6.1.4.1.9.9.999.1.5="Processing Error
 Alarm"
.1.3.6.1.4.1.9.9.999.1.6="{"alertname": "cpu-util-idle", "cluster": "unknown_cee", "hostname":
"rid7802257-brprakas-1-master3",
 "monitor": "prometheus", "replica": "unknown_cee", "severity": "critical", "instance":
"rid7802257-brprakas-1-master3",
"description": "High CPU", "type": "Processing Error Alarm"}" .1.3.6.1.4.1.9.9.999.1.7="unknown_cee"
```

```
 .1.3.6.1.4.1.9.9.999.1.8="*"
.1.3.6.1.4.1.9.9.999.1.9="rid7802257-brprakas-1-master3"
.1.3.6.1.4.1.9.9.999.1.10="rid7802257-brprakas-1-master3" .1.3.6.1.4.1.9.9.999.1.11="*"
 .1.3.6.1.4.1.9.9.999.1.12="*"
```

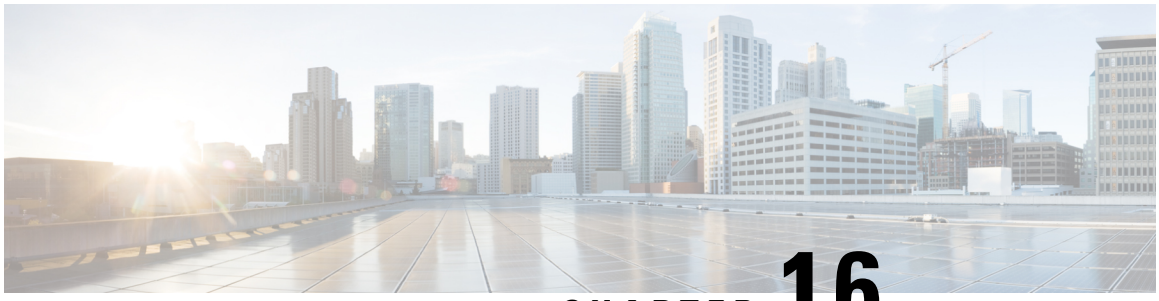


Note In alert PCAP, hostname should be present.

Step 6 Enter the `show alerts active detail` command to verify the active alerts.

Sample configuration:

```
] cee# show alerts active detail .
alerts active detail cpu-util-idle eb807845cc3d
 severity    critical
 type        "Processing Error Alarm"
 startsAt    2025-05-26T11:27:32.902Z
 source      rid7802257-brprakas-1-master3
 summary     Unknown
 labels      [ "alertname: cpu-util-idle" "cluster: unknown_cee" "hostname:
rid7802257-brprakas-1-master3" "monitor: prometheus" "replica: unknown_cee" "severity: critical" ]
 annotations [ "description: High CPU" "type: Processing Error Alarm" ]
```



CHAPTER 16

Event and System logs

- Consolidated application logs, on page 285
- Feature Description, on page 286
- How it Works, on page 286
- View the logs, on page 286
- Troubleshoot Information, on page 286
- Consolidated AAA log files, on page 287
- Configure package level logging, on page 294
- Separation of accounting retry messages logging for OCS and PMZ, on page 295
- Enhanced MongoDB pod log for debugging, on page 301
- Supplementary log forwarding for cnAAA, on page 303
- Configurable log size and PVC for consolidated logs, on page 307
- Log forwarding to external servers, on page 312
- Grafana database monitoring dashboard, on page 315
- Grafana dashboard files, on page 315
- Audit and logging enhancements, on page 316
- Log noise reduction, on page 319

Consolidated application logs

Summary Data

Table 38: Summary Data

Applicable Product(s) or Functional Area	cnAAA
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Always-on
Related Documentation	Not Applicable

Feature History

Table 39: Feature History

Feature Details	Release
First introduced.	2025.01.0

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 cnAAA (engine), cnAAA-radius-ep, policy-builder, radius-ep, crd, and unifiedapi.

View the logs

This section describes how to view the consolidated application logs.

To view the consolidated logs, use the following command:

kubectl logs -n *namespace* **consolidated-logging-0**

NOTES:

- *namespace* – Specifies the namespace under which cnAAA 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*>-cnAAA-oam-app helm deployment. To view the configured helm charts and their status, use the following command:

```
helm list
```

- Ensure that the gRPC stream appender is enabled by verifying the contents of cps-logback configMap. To verify the contents, use the following 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.

  For more information on consolidated logging, see Consolidated AAA log files section

Consolidated AAA log files

The directory `/data/consolidated-aaa-logging/`, located inside the pod `consolidated-aaa-logging-0`, contains these logs which provide a comprehensive view of system operations, user activities, API interactions, retry and failure events, and core engine and application processing.

bng-device-audit.log: Logs create, update, and delete operations on BNG devices through Policy Enforcement Point, REST API. This includes the timestamp, source IP, operation, and API content.

qns-custrefdata_audit.log: Logs create, update, and delete operations on CRD table entries through REST API. This includes the timestamp, operation, source IP, and API content.

lastlogin_user_pb.log: Logs details of the last user login to Policy Builder (PB). This includes the username, timestamp, and message.

lastlogin_user_cc.log: Logs details of the last user login to Control Center. This includes the username, timestamp, and message.

qns-audit-pb.log: Logs PB user login, logout, publish, and save client repository actions. This includes the timestamp, username, source IP, and message.

qns-audit.log: Logs Control Center user login, logout, subscriber create, update, delete, and subscriber SSID add or remove actions. This includes the timestamp, username, source IP, and message.

qns-pb-publish-svn-diff.log: Logs SVN differences when a user publishes in PB. This includes the timestamp, username, and list of repository changes.

coaBackOffFailureReport.csv: Logs CoA BackOff retry failures that exceed maximum retransmission. This includes the timestamp, session ID, BNG IP, subscriber ID, service details, error type, and NAS error code.

ocs_consolidated-retryandsuccess.csv: Logs proxy accounting messages that reached the primary or secondary OCS server after retries. This includes the timestamp, server name, server IP, retry count, and request details.

pmz_consolidated-retryandsuccess.csv: Logs proxy accounting messages that reached primary or secondary PMZ server after retries. This includes the timestamp, PMZ server name, message with PMZ server IP, number of retries, Request message details.

ocs_consolidated-retryandfailure.csv: Logs proxy accounting messages that failed to reach primary or secondary OCS server after all the retries. This includes the timestamp, OCS server name, message with OCS server IP, number of retries, Request message details.

pmz_consolidated-retryandfailure.csv: Logs proxy accounting messages that failed to reach PMZ servers after all retries. This includes the timestamp, PMZ server name, message with PMZ server IP, number of retries, Request message details.

consolidated-engine.log: Logs core engine processing messages.

consolidated-app.log: Logs application processing, including RADIUS endpoint pod, engine processing, and network device manager logs.

consolidated-api.log: Logs Unified API interactions.

policybuilder.log: Captures policy builder pod logs.

Accounting mismatch with retry and failure logging mechanism to OCS

The retry and failure logging mechanism for OCS accounting messages records accounting mismatches in CSV files when cnAAA does not deliver messages to OCS due to reachability issues.

cnAAA supports these types of Accounting messages:

- Messages that are retried and successful on successive retries will be logged in `consolidated-retryandsuccess.csv`.
- Messages that are retried and fail in all retries will be logged in `consolidated-retryandfailure.csv`.

Sample output for `consolidated-retryandsuccess.csv`

```
⟹ consolidated-retryandsuccess.csv ⟸
[beta-master-3/radius-ep/radius-ep-0] 2025-01-21 06:24:38,899  PassiveMZ-12997,Primary=false,Secondary=true,Success after r
etries for- 30.50.59.100,Number of retries = 0, Request: CALLING-STATION-ID=11-34-56-78-98-44,FRAMED-IP-ADDRESS=192.168.107
.163,NAS-PORT-TYPE=5,SERVICE-TYPE=2,USER-NAME=HA_subscriber_1051946,ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS=30.50.59.105,NAS-PORT
=99,CISCO-AVPAIR=service-name=Plan0,parent-session-id=Local_DC_1051946,ACCT-SESSION-ID=Service_Local_DC_1051946,CALLED-STAT
ION-ID=IN001_1051946,NAS-PORT-ID=4/0/2/5.75
[beta-master-3/radius-ep/radius-ep-0] 2025-01-21 06:24:38,900  PassiveMZ-12997,Primary=false,Secondary=true,Success after r
etries for- 30.50.59.100,Number of retries = 0, Request: SERVICE-TYPE=2,ACCT-STATUS-TYPE=3,NAS-IP-ADDRESS=30.50.59.105,CISC
O-AVPAIR=service-name=Plan0,parent-session-id=Local_DC_1051946,ACCT-SESSION-ID=Local_DC_1051946,CALLING-STATION-ID=11-34-56
-78-98-44,FRAMED-IP-ADDRESS=192.168.107.163,NAS-PORT-TYPE=5,ACCT-INPUT-PACKETS=67895,USER-NAME=HA_subscriber_1051946,ACCT-O
UTPUT-PACKETS=897654,ACCT-DELAY-TIME=0,NAS-PORT=2044,CALLED-STATION-ID=IN001_1051946,NAS-PORT-ID=4/0/2/5.75
```

Sample output for `consolidated-retryandfailure.csv` format.

```
⟹ consolidated-retryandfailure.csv ⟸
[beta-master-3/radius-ep/radius-ep-0] 2025-01-21 05:34:50,902  DEL_OCS,Primary=false,Secondary=false,Time out after retries
 for- 30.50.60.100,Number of retries = 0, Request: SERVICE-TYPE=2,ACCT-STATUS-TYPE=3,NAS-IP-ADDRESS=30.50.59.102,CISCO-AVPA
IR=service-name=Plan0,parent-session-id=Local_DC_553749,ACCT-SESSION-ID=Local_DC_553749,CALLING-STATION-ID=11-34-56-79-62-9
F,FRAMED-IP-ADDRESS=192.169.53.254,NAS-PORT-TYPE=5,ACCT-INPUT-PACKETS=67895,USER-NAME=HA_subscriber_553749,ACCT-OUTPUT-PACK
ETS=897654,ACCT-DELAY-TIME=0,NAS-PORT=53847,CALLED-STATION-ID=IN001_553749,NAS-PORT-ID=4/0/2/5.75
[beta-master-3/radius-ep/radius-ep-0] 2025-01-21 05:34:50,902  DEL_OCS,Primary=false,Secondary=false,Time out after retries
 for- 30.50.60.100,Number of retries = 0, Request: SERVICE-TYPE=2,ACCT-STATUS-TYPE=3,NAS-IP-ADDRESS=30.50.59.102,CISCO-AVPA
IR=service-name=Plan0,parent-session-id=Local_DC_554786,ACCT-SESSION-ID=Local_DC_554786,CALLING-STATION-ID=11-34-56-79-66-A
C,FRAMED-IP-ADDRESS=192.169.58.11,NAS-PORT-TYPE=5,ACCT-INPUT-PACKETS=67895,USER-NAME=HA_subscriber_554786,ACCT-OUTPUT-PACKE
TS=897654,ACCT-DELAY-TIME=0,NAS-PORT=54884,CALLED-STATION-ID=IN001_554786,NAS-PORT-ID=4/0/2/5.75
```

Log file location in `consolidated-aaa-logging-0 pod：/data/consolidated-aaa-logging.`

```
-rw-rw-r-- 1 303 303 143M Jan 21 05:34 consolidated-retryandfailure.csv
-rw-r--r-- 1 303 303  51M Jan 21 06:24 consolidated-retryandsuccess.csv
I have no name!@consolidated-aaa-logging-0:/data/consolidated-aaa-logging$
```

System and Syslog activity logging

System and Syslog activity logging allows you to track and monitor user activities within the CPC environment. This feature provides data for security auditing and troubleshooting by capturing logs of user sessions, command executions, and system events.

This feature helps you:

- Monitor for unauthorized access or suspicious activity by tracking login attempts and command history.

- Identify system changes or specific commands that lead to operational issues such as deletion of files, change in access control or system configuration changes.
- Maintain a chronological record of user interactions with the system.

The logging framework uses standard Linux utilities and the `psacct` package (integrated as the `userlog` tool) to provide:

- **User session tracking**: The system records login and logout times, session durations, and source IP addresses.
- **Command history**: Captures a list of commands executed by specific users or across the entire system.
- **System reboot history**: Tracks the frequency and timestamps of system reboots.
- **Security monitoring**: Logs failed user login attempts to identify potential brute-force attacks.

Command reference and usage

Use the commands described in this section to retrieve activity logs. Some commands require `sudo` privileges.

- **Monitor user sessions**:

  The `last` command displays a list of all users who logged in and out since the log file was created.

  Example output:

```
user@rid8447988-1-master1:~$ sudo last
luser    pts/0        10.0.1.20        Tue Feb 24 07:27   still logged in
luser    pts/0        10.0.1.20        Tue Feb 24 05:59 - 06:24  (00:25)
luser    pts/0        10.0.1.20        Tue Feb 24 04:24 - 05:46  (01:22)
reboot   system boot  6.8.0-58-generic Mon Feb 23 21:21   still running
reboot   system boot  6.8.0-58-generic Mon Feb 23 10:12   still running
reboot   system boot  6.8.0-58-generic Fri Feb 20 21:25   still running
luser    pts/0        10.0.1.20        Fri Feb 20 08:42 - 08:57  (00:15)
luser    pts/0        10.0.1.20        Fri Feb 20 04:22 - 05:02  (00:39)
luser    pts/0        10.0.1.20        Thu Feb 19 14:04 - 14:20  (00:15)
luser    pts/0        10.0.1.20        Thu Feb 19 13:57 - 14:01  (00:04)
luser    pts/1        10.0.1.20        Thu Feb 19 12:25 - 13:34  (01:09)
luser    pts/0        10.0.1.20        Thu Feb 19 12:00 - 13:26  (01:26)
luser    pts/0        10.0.1.20        Thu Feb 19 11:06 - 11:59  (00:52)
luser    pts/1        10.0.1.20        Thu Feb 19 10:52 - 11:36  (00:43)
luser    pts/0        10.0.1.20        Thu Feb 19 10:46 - 11:02  (00:15)
luser    pts/2        10.0.1.20        Thu Feb 19 10:35 - 10:51  (00:15)
luser    pts/1        10.0.1.20        Thu Feb 19 10:32 - 10:47  (00:15)
luser    pts/0        10.0.1.20        Thu Feb 19 10:26 - 10:46  (00:19)
luser    pts/0        10.0.1.20        Thu Feb 19 10:23 - 10:24  (00:00)
luser    pts/0        10.0.1.20        Thu Feb 19 10:21 - 10:22  (00:00)
luser    pts/0        10.0.1.20        Thu Feb 19 09:53 - 10:21  (00:27)
luser    pts/0        10.0.1.20        Thu Feb 19 09:35 - 09:51  (00:15)
luser    pts/0        10.0.1.20        Thu Feb 19 06:13 - 06:30  (00:16)
luser    pts/0        10.0.1.20        Thu Feb 19 04:37 - 04:54  (00:16)
```

- **Identify last login per user**:

  The `lastlog` command displays the login name, port, and the last time each user (including system users like root and daemon) logged into the system.

  Example output:

```
luser@rid8447988-1-master1:~$ sudo lastlog
Username         Port     From                                       Latest
root                                                               **Never logged in**
```

```
daemon                                                          **Never logged in**
bin                                                             **Never logged in**
sys                                                             **Never logged in**
sync                                                            **Never logged in**
games                                                           **Never logged in**
man                                                             **Never logged in**
lp                                                              **Never logged in**
mail                                                            **Never logged in**
news                                                            **Never logged in**
uucp                                                            **Never logged in**
proxy                                                           **Never logged in**
www-data                                                        **Never logged in**
backup                                                          **Never logged in**
list                                                            **Never logged in**
irc                                                             **Never logged in**
_apt                                                            **Never logged in**
nobody                                                          **Never logged in**
systemd-network                                                 **Never logged in**
systemd-timesync                                                **Never logged in**
dhcpcd                                                          **Never logged in**
messagebus                                                      **Never logged in**
systemd-resolve                                                 **Never logged in**
pollinate                                                       **Never logged in**
polkitd                                                         **Never logged in**
syslog                                                          **Never logged in**
uuidd                                                           **Never logged in**
tcpdump                                                         **Never logged in**
tss                                                             **Never logged in**
landscape                                                       **Never logged in**
fwupd-refresh                                                   **Never logged in**
usbmux                                                          **Never logged in**
sshd                                                            **Never logged in**
_aide                                                           **Never logged in**
systemd-coredump                                                **Never logged in**
_chrony                                                         **Never logged in**
sssd                                                            **Never logged in**
tac                                                             **Never logged in**
luser            pts/0    10.0.1.20                             Tue Feb 24 07:27:28
 +0000 2026
luser@rid8447988-vbagalak-1-master1:~$
```

- **View command history**:

  The `history` command displays the buffer of the most recent commands entered in the current session.

  Example output:

```
luser@rid8447988-1-master1:~$ history
   23  2025-12-08 13:16:56 ls
   24  2025-12-08 13:16:59 cat global-smi.yaml
   25  2025-12-08 13:17:00 ;s
   26  2025-12-08 13:17:03 cat cee-smi.yaml
   27  2025-12-08 13:22:02 vim cee-smi.yaml
   28  2025-12-08 13:24:39 ls
   29  2025-12-08 13:24:43 cat global-smi.yaml
   30  2025-12-08 13:33:13 cat cee-smi.yaml
   31  2025-12-09 12:14:26 kubectl get pods -n pcf | grep prom
   32  2025-12-09 12:14:43 kubectl exec -it prometheus-rules-etcd-5cf4d58496-wfpnb -n
cpf -- bash
   33  2025-12-09 12:14:50 kubectl exec -it prometheus-rules-etcd-5cf4d58496-wfpnb -n
pcf -- bash
   34  2025-12-10 05:46:25 ssh -p 2024 admin@10.100.133.79
   35  2025-12-10 05:54:51 kubectl get pods -A | grep redis
   36  2025-12-10 05:55:13 kubect logs redis-keystore-0 -n pcf
   37  2025-12-10 05:55:23 kubectl logs redis-keystore-0 -n pcf
```

```
38  2025-12-10 05:55:31 kubectl logs redis-keystore-1 -n pcf
39  2025-12-10 05:55:44 kubectl logs redis-queue-0 -n pcf
40  2025-12-10 05:55:49 kubectl logs redis-queue-1 -n pcf
41  2025-12-10 05:55:54 kubectl logs redis-queue-2 -n pcf
42  2025-12-10 05:56:02 ssh -p 2024 admin@10.100.133.79
43  2025-12-10 05:59:10 kubectl exec -it redis-queue-0 -n pcf -- bash
44  2025-12-10 06:00:02 ssh -p 2024 admin@10.100.133.79
45  2025-12-10 06:01:23 kubectl exec -it redis-queue-0 -n pcf -- bash
46  2025-12-10 06:02:10 kubectl get pods -A | grep redis
47  2025-12-10 08:08:36 kubectl get pods -A | grep aaa
48  2025-12-10 08:08:47 kubectl get pods -A -o wide | grep aaa
49  2025-12-10 08:09:00 ip r
50  2025-12-10 08:09:18 ssh -i luser.pem luser@10.1.43.220
51  2025-12-10 08:09:39 kubectl get nodes -o wide
52  2025-12-10 08:09:51 ssh -i luser.pem luser@10.1.44.174
53  2025-12-10 08:10:32 cd /data/
```

- **Track specific user activity**:

  The `.bash_history` command shows all commands a specific user executed over time, inspect the user's bash history file.

  Example output:

```
luser@rid8447988-1-master1:~$ sudo cat /home/luser/.bash_history
kubectl exec -it redis-queue-0 -n pcf -- bash
#1764911584
ssh -p 2024 admin@10.100.133.79
#1764912019
kubectl get pods -n <namespace>
#1764912023
kubectl get pods -n pcf
#1764912041
kubectl get pods -n pcf | grep redis
#1764912071
kubectl logs redis-queue-1 -n pcf
#1764912088
kubectl logs redis-queue-0 -n pcf
#1764912100
kubectl get pods -n pcf | grep redis
#1764912119
kubectl logs redis-queue-1 -n pcf
#1764912125
kubectl get pods -n pcf | grep redis
#1764912141
ssh -p 2024 admin@10.100.133.79
```

- **View system reboot history**:

  The `last reboot` command tracks how many times the system rebooted and the kernel version used.

  Example output:

```
luser@rid8447988-1-master1:~$ sudo last reboot
reboot   system boot  6.8.0-58-generic Mon Feb 23 21:21   still running
reboot   system boot  6.8.0-58-generic Mon Feb 23 10:12   still running
reboot   system boot  6.8.0-58-generic Fri Feb 20 21:25   still running
reboot   system boot  6.8.0-58-generic Wed Feb  4 04:39   still running
```

- **View failed login attempts**:

  The `lastb` command lists all failed login attempts recorded in the `/var/log/btmp` file.

  Example output:

```
luser@rid8447988-1-master1:~$ sudo lastb
vagrant  ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
service  ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
service  ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
nessus_7 ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
service  ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
Fortiman ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
nessus_7 ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
nessus_7 ssh:notty    173.37.95.247    Sun Feb 22 16:30 - 16:30  (00:00)
         ssh:notty    173.37.95.247    Sun Feb 22 16:25 - 16:25  (00:00)
cisco    ssh:notty    173.37.95.247    Sun Feb 22 16:25 - 16:25  (00:00)
cisco    ssh:notty    173.37.95.247    Sun Feb 22 16:25 - 16:25  (00:00)
cisco    ssh:notty    173.37.95.247    Sun Feb 22 16:25 - 16:25  (00:00)
ADSL     ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
ADSL     ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
ADSL     ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
adminuse ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
adminuse ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
adminuse ssh:notty    173.37.95.247    Sun Feb 22 16:16 - 16:16  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
Administ ssh:notty    173.37.95.247    Sun Feb 22 16:15 - 16:15  (00:00)
```

Search Facility

The `search_facility.sh` script searches and filters large volumes of compressed consolidated engine log files for cnAAA. This utility automates troubleshooting network accounting issues, identifies subscriber sessions, and analyze network traffic patterns by correlating MAC addresses with session IDs and message types within specific time ranges. The primary goal is to reduce log analysis time, enhancing operational efficiency.

How the log search utility works

The `Search_Facility.sh` script, found in the `pcf-utilities` pod, is copied and executed from the log directory on a remote logging server.

- It processes compressed .gz log files that follow a standard date-time naming convention.
- The script supports searching for ServiceAccounting and SessionAccounting messages.
- It correlates MAC addresses with session IDs across these message types.
- The script filters logs by specific time ranges, MAC addresses, and session IDs.
- All search results compile into a single output file.
- The script features interactive prompts with error handling for user inputs.

Log forwarder utility

The `logs_forwarder.sh` Bash script automates transferring log files from a local directory to a remote server using SCP. It renames each file with a time-stamp based on its last modification time for version control and chronological organization.

Use the Log Forwarder Utility: Follow these steps to use the log forwarder script.

1. Locate the `logs_forwarder.sh` script in the `pcf-utilities-0` pod under `/data/utilities/support/script/search-facility/`.
2. Copy the `logs_forwarder.sh` script to the location where the log files are available.
3. Modify the `logs_forwarder.sh` file by providing the following details:
   - `SRC_DIR`: Source directory containing log files to transfer.
   - `DEST_DIR`: Destination directory on the remote server.
   - `DEST_HOST`: IP address of the destination server.
   - `DEST_USER`: Username for remote server authentication.
4. Execute the script to forward logs.

   ```
   ./logs_forwarder.sh
   ```

The script renames the log files by appending the time-stamp (consolidated-engine-YYYY-MM-DD-HH.X.log.gz).



Note The `logs_forwarder.sh` script is for reference and can be modified according to specific requirements.

Use the log search utility

Procedure

Follow these steps to use the log search utility:

Step 1 Log in to a cnAAA server.

Step 2 Locate the `search_facility.sh` script in the `pcf-utilities-0` pod under `/data/utilities/support/script/search-facility/`

Step 3 Copy the `search_facility.sh` script to the specific log directory on the remote logging server where the search is performed.

Step 4 Open a terminal in the log directory.

Step 5 Execute the script, providing the log directory path as an argument.

```
./search_facility.sh /path/to/logs
```

Replace `/path/to/logs` with the actual log directory path.

Step 6 When prompted by the script, enter these parameters:

a) **Start Date**: Enter the date in yyyy-mm-dd format.
b) **Start/End Hour** (Optional): Specify hours in 00-23 format. If a start hour is provided, the end hour is mandatory.
c) **End Date**: Enter the date in yyyy-mm-dd format.
d) **MAC Address**: Enter the MAC address in xxxx.xxxx.xxxx format.
e) **Message Type** (Optional): Specify ServiceAccounting or SessionAccounting. If omitted, both types are included by default.

f) **Session ID** (Optional): Provide the Acct-Session-Id for session correlation.

Note The script validates entries for correct formats, logical consistency (for example, no future dates, valid MAC address, hour ranges), and required arguments. If an entry is invalid, the script prompts again or exits with an informative message.

Example

Running the Search Facility Script

```
root@m6-calipers-vm3:~# ./search_facility.sh /root/search_facility_logs
Log Directory /root/search_facility_logs
Enter Start Date(yyyy-mm-dd): 2025-09-14
Enter Start Hour(00..23) [Optional]: 00
Enter End Date(yyyy-mm-dd): 2025-09-14
Enter End Hour(00..23): 23
Enter Mac Address(xxxx.xxxx.xxxx): 7B11.9A3C.0005
Enter Message Type(ServiceAccounting/SessionAccounting) [Optional]:
INFO: No message type provided. Both ServiceAccounting and SessionAccounting will be used
as search criteria.
********************************************** Session ID
**********************************************
Enter Session ID [Optional]: 00604beb
********************************************** Final Result
**********************************************
Final output gets stored in file named search_results present under /root/search_facility_logs
************************************************************************************************************
```

Configure package level logging

Configure core packages to log only error-level messages to filters out high-volume or debug data. This reduces log noise, optimizes Persistent Volume Claim (PVC) storage, and minimizes I/O overhead. This configuration helps to identify system errors quickly by generating relevant log entries.

Follow these steps to configure package level logging:

Procedure

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

Step 2 Set the logging level for the core Cisco package.

Example:

```
debug logging logger com.cisco
  level error
exit
```

Step 3 Set the logging level for the RADIUS package.

```
debug logging logger com.cisco.radius
  level error
exit
```

Step 4 `commit` the changes.

Separation of accounting retry messages logging for OCS and PMZ

Feature description

Feature Name	Release Information	Description
Separation of accounting retry messages logging for OCS and PMZ	2025.03.0	This feature enables a separate accounting retry message logs for Online Charging System (OCS) servers and Passive/Offline Charging System (PMZ) servers. By adding a server-type parameter in the Ops Center configuration, the system organizes logs based on server type. This update improves log traceability and troubleshooting by distinguishing between success and failure logs. This enhances operational efficiency and system management.

This feature separates accounting retry message logs for OCS and PMZ servers. The system creates dedicated log files for each server type and further classifies them into "retry success" and "retry failure" categories. By configuring a new server-type property, the system directs logs to the appropriate files. This structured logging approach simplifies troubleshooting and improves clarity in log data management, making it easier to identify and resolve issues.

How this feature works

- The system adds a server-type parameter to the RADIUS server configuration. Set "online" for OCS or "offline" for PMZ.
- On each accounting retry, the system checks the server-type and writes logs to the corresponding OCS or PMZ files.
- Each server type uses two log files: one for retry successes and one for retry failures.
- If you do not set the server-type, the system defaults to OCS (online) log files.
- The system manages log files with automatic rotation based on size and file count.

Configuration

Procedure

Step 1 Set the `server-type` for each RADIUS server.

Example:

```
radius server-group grp1
 servers WB_12100
  server-type online
exit

servers PassiveMZ-12998
  server-type offline
exit
```

Note Assign `server-type online` for OCS servers and `server-type offline` for PMZ servers to ensure logs are categorized correctly.

Note Ensure CEE namespace is deployed to support Persistent Volume.

Step 2 Enable persistent storage to retain the log files.

```
pcf(config)# k8s use-volume-claims true
pcf(config)# db global-settings volume-storage-class local
```

Step 3 Ensure **consolidated-aaa-logging-0** pod is not added under disable-pods.

```
pcf(config)# show full-configuration pods-management disable-pods
% No entries found.
```

Step 4 Verify the log file updates according to the configured server -type.

```
kubectl exec -it -n pcf consolidated-aaa-logging-0 -- bash
cd /data/consolidated-aaa-logging/
ls
ocs_consolidated-retryandsuccess.csv
ocs_consolidated-retryandfailure.csv
pmz_consolidated-retryandsuccess.csv
pmz_consolidated-retryandfailure.csv
```

Use cases

These use cases show the configuration for logging of proxy accounting retry messages for OCS and PMZ servers.

No log for successful first attempt: The system does not generate a log entry when an accounting message successfully reaches the primary OCS or PMZ server on the first attempt.

```
radius server-group grp1
  servers WB_12100
    primary    10.1.43.142
    secondary  10.1.43.140
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type online
    retries    3
  exit

  servers PassiveMZ-12998
    primary    10.1.45.171
    secondary  10.1.45.170
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type offline
    retries    3
  exit
```

Log for failure after all retries: The system generates a log entry in `ocs_consolidated-retryandfailure.csv`,`pmz_consolidated-retryandfailure.csv` when an accounting message fails to reach both primary and secondary OCS or PMZ servers after all retries.

```
radius server-group grp1
  servers WB_12100
    primary    10.1.43.142
    secondary  10.1.43.140
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type online
    retries    3
  exit

  servers PassiveMZ-12998
    primary    10.1.45.171
    secondary  10.1.45.170
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type offline
    retries    3
  exit
```

This is a sample log entry for OCS server in **log for failure after all retries scenario.**

```
2025-06-16 18:13:25,940  WB_12100,Primary=false,Secondary=false,Time out after retries for-
 10.1.43.140,
Number of retries = 3, Request:
FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.2A02,NAS-PORT-TYPE=5,
USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.2A02,ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS=11.11.118.69,
NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

This is a sample log entry for PMZ server in **log for failure after all retries scenario.**

```
2025-06-16 18:13:15,960  PassiveMZ-12998,
Primary=false,Secondary=false,Time out after retries for- 10.1.45.170,Number of retries =
3,
Request: FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.2A02,
NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.2A02,ACCT-STATUS-TYPE=1,
NAS-IP-ADDRESS=11.11.118.69,NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,
service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

Log for success on secondary after primary fails: The system generates a log entry in `ocs_consolidated-retryandsuccess.csv`, `pmz_consolidated-retryandsuccess.csv` when an accounting message fails to reach the primary, but reaches the secondary OCS or PMZ server after all retries.

```
radius server-group grp1
  servers WB_12100
    primary    10.1.43.142
    secondary  10.1.43.140
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type online
    retries    3
  exit

  servers PassiveMZ-12998
    primary    10.1.45.171
    secondary  10.1.45.170
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type offline
    retries    3
  exit
```

This is a sample log entry for OCS server in **Log for success on secondary after primary fails scenario**.

```
2025-06-17 09:01:36,934  WB_12100,Primary=false,
Secondary=true,Success after retries for- 10.1.43.140,
Number of retries = 0, Request: FRAMED-IP-ADDRESS=192.168.3.10,
CALLING-STATION-ID=0005.9A3C.1A01,NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,
USER-NAME=0005.9A3C.1A01,ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS=11.11.118.69,
NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,service-name=A0F0050M050M000005MQ,
portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

This is a sample log entry for PMZ server in **Log for success on secondary after primary fails scenario**.

```
2025-06-17 09:01:38,013  PassiveMZ-12998,Primary=false,
Secondary=true,Success after retries for- 10.1.45.170,Number of retries = 0,
 Request: FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.1A01,
NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.1A01,
ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS=11.11.118.69,NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,
service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

Log for success after retries to primary: The system generates a log entry in `ocs_consolidated-retryandsuccess.csv`, `pmz_consolidated-retryandsuccess.csv` when an accounting message is delivered to the primary OCS or PMZ server after one or more retries.

```
radius server-group grp1
  servers WB_12100
    primary    10.1.43.142
    secondary  10.1.43.140
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type online
    retries    3
  exit

  servers PassiveMZ-12998
    primary    10.1.45.171
    secondary  10.1.45.170
    nas-ip     11.11.118.69
    timeout-seconds      5
    server-type offline
```

```
    retries    3
  exit
```

This is a sample log entry for OCS server in **Log for success after retries to primary scenario**.

```
2025-06-13 18:52:25,632 WB_12100,Primary=true,Secondary=false,
Success not at first time - for- 10.1.43.142,Number of retries = 2,
 Request: FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.4A04,
NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.4A04,
ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS=
11.11.118.69,NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,
service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

This is a sample log entry for PMZ server in **Log for success after retries to primary scenario**.

```
2025-06-13 19:06:08,883 PassiveMZ-12998,Primary=true,
Secondary=false,Success not at first time - for- 10.1.45.171,
Number of retries = 2, Request: FRAMED-IP-ADDRESS=192.168.3.10,
CALLING-STATION-ID=0005.9A3C.4A04,NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,
USER-NAME=0005.9A3C.4A04,ACCT-STATUS-TYPE=1,NAS-IP-ADDRESS= 11.11.118.69,
NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

Log for full queue: The system generates a log entry when the proxy queue is full and cannot process additional messages.

```
radius server-group grp1
  servers WB_12100
    primary    10.1.43.142
    secondary  10.1.43.140
    nas-ip     11.11.118.69
    timeout-seconds      5
    thread-pool-size 300
    server-type online
    max-proxy-queue-size 50000
    retries    3
  exit

  servers PassiveMZ-12998
    primary    10.1.45.171
    secondary  10.1.45.170
    nas-ip     11.11.118.69
    timeout-seconds      5
    thread-pool-size 300
    server-type offline
    max-proxy-queue-size 50000
    retries    3
  exit
```

This is a sample log entry for OCS server in **Log for full queue scenario**.

```
2025-07-05 14:59:10,109 WB_12100,Primary=false,
Secondary=false,Queue Full for - 10.1.43.142,Number of retries = 0,
 Request: FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.5A05,
NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.5A05,ACCT-STATUS-TYPE=3,
NAS-IP-ADDRESS=11.11.118.69,NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,
service-name=A0F0050M050M000005MQ,portbundle=enable,ACCT-SESSION-ID=child_00011,
NAS-PORT-ID=0/0/0/701
```

This is a sample log entry for PMZ server in **Log for full queue scenario**.

```
2025-07-05 14:59:10,108 PassiveMZ-12998,Primary=false,
 Secondary=false,Queue Full for - 10.1.45.171,Number of retries = 0,
Request: FRAMED-IP-ADDRESS=192.168.3.10,CALLING-STATION-ID=0005.9A3C.5A05,
NAS-PORT-TYPE=5,USER-PASSWORD=xxxxxxxxx,USER-NAME=0005.9A3C.5A05,
ACCT-STATUS-TYPE=3,NAS-IP-ADDRESS=11.11.118.69,NAS-PORT=0,CISCO-AVPAIR=parent-session-id=00011,
```

```
service-name=A0F0050M050M000005MQ,portbundle=enable,
ACCT-SESSION-ID=child_00011,NAS-PORT-ID=0/0/0/701
```

This is a sample log file for **Grafana GUI user login details**:

```
Grafana user logging details are captured in Grafana pod which will be present under CEE
namespace
$ kubectl get pods -A|grep graf
cee               grafana-bc8df4c5d-zxhb6                                    2/2
Running   3 (16d ago)     28d
cee               grafana-dashboard-metrics-9d666f59-m67zg                   1/1
Running   0               28d
pcf               grafana-dashboard-cdl-pcf-764ccc76f9-l29jd                 1/1
Running   0               16d
pcf               grafana-dashboard-etcd-pcf-77664d6867-27fvb                1/1
Running   0               16d
pcf               grafana-dashboard-pcf-748fb659bd-9sfqj                     1/1
Running   0               16d
Luser@rid8043277-aellendu-1-master1:~$

NOTE: The above Grafana pods logs user logging details.

Sample Logs:
-->If Grafana httproutes is opened in browser:

logger=context userId=0 orgId=0 uname= t=2025-10-24T11:10:29.611429757Z level=info
msg="Request Completed" method=GET path=/ status=302 remote_addr=10.1.14.204 time_ms=0
duration=479.628µs size=29 referer= handler=/ status_source=server

-->If logged in with admin credentials

logger=context userId=2 orgId=1 uname=admin t=2025-10-24T11:13:54.213746926Z level=info
msg="Request Completed" method=GET path=/api/live/ws status=-1 remote_addr=11.11.76.195
time_ms=20 duration=20.484741ms size=0 referer= handler=/api/live/ws status_source=server

-->If admin user logout is performed

logger=http.server t=2025-10-24T11:15:39.54041156Z level=info msg="Successful Logout"
userID=2
logger=context userId=2 orgId=1 uname=admin t=2025-10-24T11:15:39.540592971Z level=info
msg="Request Completed" method=GET path=/logout status=302 remote_addr=11.11.76.195 time_ms=53
 duration=53.590989ms size=29 referer="https://grafana.10.86.70.223.nip.io/?orgId=1"
handler=/logout status_source=server
```

Enhanced MongoDB pod log for debugging

Feature history

Feature Name	Release Information	Description
Enhanced MongoDB pod logging for debugging	2026.01.0	This feature enhances MongoDB pod logging within the Cisco Policy Controller (CPC) application to address insufficient log retention, excessive disk consumption, and inefficient debugging in production environments. It manages local pod logs with rotation, and forwards logs to external servers for centralized analysis and long-term archival. This approach significantly improves troubleshooting capabilities and overall observability for operational teams.

This feature enhances MongoDB pod logging within the CPC application. It resolves critical issues caused by insufficient log retention in production environments. In the current setup, logs are only available for a few hours, which is inadequate for effective debugging and root cause analysis. This enhancement addresses three key problems: excessive disk space consumption caused by unmanaged logs, debugging inefficiency due to limited log data, and the lack of robust centralized observability needed for comprehensive analysis and archival.

How MongoDB log forwarding works

MongoDB manages logs through both local log management and external forwarding:

- **Local pod log management and on-node access**
  - MongoDB application logs are generated and managed locally within each pod's filesystem, which helps prevent logs from consuming excessive disk space within the pod.
  - Systemd Journald collects all logs generated by pods on a Kubernetes node and acts as an intermediate storage layer on the node.
  - Systemd Journald is configured to allocate a portion of the node's disk space for log storage, allowing for a significantly longer retention period typically 2 to 3 days of log history on the node, depending on disk size and log volume.
  - On-node storage is crucial for immediate debugging. It allows operational teams to access logs directly on the affected node without relying solely on external systems.
- **External log forwarding for centralized analysis**
  - Once Systemd Journald collects logs, they are reliably and efficiently transferred to external log servers. This process integrates with the existing CEE log forwarding infrastructure.

- Logs are distributed to several external logging destinations, including platforms such as Splunk, Fluentd, and syslog. This enables long-term archival, advanced analytics, and centralized monitoring across the entire environment.
- Log forwarding supports various transport protocols for secure transmission.

Accessing MongoDB logs

MongoDB logs can be accessed by using two methods:

- **On-node debugging**: Retrieve logs directly from the Kubernetes node where your MongoDB pod is running.
  - Use `crictl` to list all containers and filter for the MongoDB pod.

    ```
    sudo crictl ps -a | grep "mongo "
    ```

  - Use `journalctl` to retrieve logs for the specific container.

    ```
    sudo journalctl UNIT=cri CONTAINER_ID=<trimmed_container_id>
    --directory=/var/log/journal --output=short-iso-precise
    ```



Note When using `journalctl`, the logs include additional metadata, such as the node name and the container ID. This provides more context compared to raw pod logs.

- **Centralized log analysis**: For long-term archival, advanced searching, and correlation across multiple services, forward logs to external systems.
  - Centralized logging platforms such as Splunk, Fluentd dashboards, or syslog servers provide powerful capabilities for search, filtering, visualization, and alerting.
  - For more information about forwarding logs to external systems, see the section Log forwarding to external servers.

Known limitations

- Variable log retention: The duration that logs are retained on a node (for example, 2 to 3 days) depends on the volume of logs generated by the applications and the total disk space that Journald can use. A high volume of traffic or verbose logging reduces the retention period.
- No user interface changes: This feature involves backend logging infrastructure enhancements. The application does not introduce new user interface elements, dashboards, or configuration options.

Supplementary log forwarding for cnAAA

Feature history

Feature Name	Release Information	Description
Supplementary log forwarding for cnAAA	2026.01.0	The cnAAA feature automates log forwarding for critical logs. It identifies these logs, collects them, formats the entries, and transfers them securely to a remote server using scheduled cron jobs and password-less SSH.

This feature introduces an automated and secure log forwarding mechanism for the `consolidated-aaa-logging` pod to ensure complete log collection. It addresses instances where primary log forwarding methods, such as Fluent Bit or syslog, might miss critical engine or audit logs. The system transfers all the compressed consolidated log files *(consolidated-engine-*log.gz*, *consolidated-api-*log.gz*, *consolidated-app-*log.gz ocs_consolidated-retryandsuccess*.csv.gz*, *ocs_consolidated-retryandfailure*.csv.gz*) to a designated remote server using scheduled cron jobs and passwordless SSH key-based authentication. This provides essential data for troubleshooting and compliance.

Log forwarding mechanism

The log forwarding mechanism uses the `log-copy-startup.sh` script that performs these actions:

- **Log identification**: Scans the configured source directory for all the compressed consolidated log files *(consolidated-engine-*log.gz*, *consolidated-api-*log.gz*, *consolidated-app-*log.gz ocs_consolidated-retryandsuccess*.csv.gz*, *ocs_consolidated-retryandfailure*.csv.gz*).
- **Log transfer**: The script uses a transfer cache to identify and securely transfer only new or modified logs to a specified remote log server, preventing duplicate transfers.
- **Passwordless authentication**: The system automatically generates an RSA SSH key pair (`/home/appuser/.ssh/id_rsa`).

Note To enable secure, passwordless SCP transfers, manually copy the public key to the destination server during the first execution.

- **Automation**: The transfer process can be fully automated using scheduled cron jobs, enabling transfers at defined intervals.
- **Error handling**: Manages errors during staging and transfer and logs them for troubleshooting.
- **Operational logging**: Logs all scheduled and manual executions to `data/consolidated-aaa-logging/external-scp-copy.log`. This file automatically rotates to maintain a size under 10 MB by retaining the last 1000 lines.

(Optional) Cron job automation

Automated transfers are disabled by default. Use these parameters to enable and schedule transfers:

- **enable-cronjob {true | false**: By default, this automation is disabled. To enable it, set the `enable_cronjob` parameter in the Ops-Center CLI.



Note It is recommended to configure an explicit schedule when enabling the cron job. If you set `enable-cronjob` to `true` without defining `cron-schedule` values, the log copy operation runs every minute.

- **cron-schedule minute**: Specifies the hour field for the cron schedule. The range is 0 to 23. For example, use 2 for 2 AM.
- **cron-schedule day-of-month** : Specifies the day of the month for the cron schedule. The range is 1 to 31.
- **cron-schedule month**: Specifies the month for the cron schedule. The range is 1 to 12.
- **cron-schedule day-of-week**: Specifies the day of the week for the cron schedule. The range is 0 to 6, where 0 is Sunday.

Configure log forwarding mechanism in Ops-Center

Before you begin

Complete these manual steps on the destination server before the log forwarding mechanism can function.

1. Configure the SSH port on the destination server:
   a. Edit the SSH daemon configuration file:
      ```
      sudo vi /etc/ssh/sshd_config
      ```
   b. Add or modify the Port directive (e.g., Port 2222).
   c. Restart the SSH service:
      ```
      sudo systemctl restart sshd
      ```
2. Copy the public SSH key generated by the `log-copy-startup.sh` script to the `authorized_keys` file on the destination server. This enables password-less transfers. This is a one-time setup.
   a. The script generates the key pair at */home/appuser/.ssh/id_rsa* (private) and */home/appuser/.ssh/id_rsa.pub* (public).
   b. Use the ssh-copy-id command from the AAA logging pod to copy the public key:
      ```
      sudo ssh-copy-id -i /home/appuser/.ssh/id_rsa.pub -p <DEST_PORT>
      <DEST_USER>@<DEST_HOST>
      ```
      ```
      kubectl exec -it consolidated-aaa-logging-0 -n <CPC-NAMEPSACE> -- bash
      ```
   c. The script displays setup instructions in the `external-scp-copy.log` file when it generates a new key. The key persists across pod restarts because the system stores it in a PVC.

If you use a port other than the default port 22, ensure that the port is open on the destination server.

Follow these steps to configure the log forwarding mechanism:

Procedure

Step 1 Log in to the cnAAA Ops-Center and enter the `config` command.

Step 2 Set the destination server details.

- Set the destination IP address

  ```
  debug logging external-scp-server dest-ip <dest-ip>
  ```

- Set the directory path on the remote server where logs will be stored.

  ```
  debug logging external-scp-server dest-path <dest-path-on-remote-server>
  ```

- Set the site name of the remote server.

  ```
  debug logging external-scp-server dest-site-name <site-name>
  ```

- Set the destination port of the remote server.

  ```
  debug logging external-scp-server dest-port <port>
  ```

- Set the destination user-name.

  ```
  debug logging external-scp-server dest-username <username>
  ```

Step 3 Define the source log directory and SSH key.

- Set the local directory path on the cnAAA node that contains the logs to transfer.

  ```
  debug logging external-scp-server source-directory <source-path-of-logs>
  ```

Step 4 Enable or disable the automated scheduled execution of log transfers. It is disabled by default.

- To enable

  ```
  debug logging external-scp-server enable-cronjob true
  ```

- To disable

  ```
  debug logging external-scp-server enable-cronjob false
  ```

If enabled, define the cron expression to schedule the log transfer frequency.

```
debug logging external-scp-server cron-schedule minute <minute>
debug logging external-scp-server cron-schedule hour <hour>
debug logging external-scp-server cron-schedule day-of-month <day-of-month>
debug logging external-scp-server cron-schedule month <month>
debug logging external-scp-server cron-schedule day-of-week <day-of-week>
```

Step 5 Commit the changes to apply the configuration.

```
commit
```

Step 6 Verify the log forwarding configuration.

```
show running-config debug logging external-scp-server
```

Example:

Configurations without a cron job.

```
pcf(config)# debug logging external-scp-server dest-ip 192.0.2.100
pcf(config)# debug logging external-scp-server dest-port 22
pcf(config)# debug logging external-scp-server dest-username scpuser
pcf(config)# debug logging external-scp-server dest-path /remote/logs
pcf(config)# debug logging external-scp-server dest-site-name site_3
pcf(config)# debug logging external-scp-server source-directory /var/logs/local
pcf(config)# debug logging external-scp-server enable-cronjob false
```

Example:

Configuration for every 2 minutes: This schedules transfers every 2 minutes.

```
pcf(config)# debug logging external-scp-server dest-ip 192.0.2.121
pcf(config)# debug logging external-scp-server dest-port 2222
pcf(config)# debug logging external-scp-server dest-username root
pcf(config)# debug logging external-scp-server dest-path /root
debug logging external-scp-server dest-site-name site_3
pcf(config)# debug logging external-scp-server source-directory /data/consolidated-aaa-logging/
pcf(config)# debug logging external-scp-server enable-cronjob true
pcf(config)# debug logging external-scp-server cron-schedule minute */2
```

Remote log directory structure

The system stores consolidated log files in specific directories on the remote server based on the site origin and log type.

- Consolidated API, engine, and application logs:

  ```
  <destination-path>/<site-name>/<api/app/engine>/<year>/<month>
  ```

- Consolidated OCS retry logs:

  ```
  <destination-path>/<site-name>/<ocs_retry_failure/ocs_retry_success>
  ```

Note If the volumes are available on the source server, the system performs an immediate log file copy as soon as the cron job starts.

Configurable log size and PVC for consolidated logs

Feature history

Feature Name	Release Information	Description
Configurable size, index and PVC for logs	2026.01.0	This cnAAA feature provides flexible log management for `consolidated-aaa-logging-0` pods. Administrators can use the Ops-Center CLI to configure log size, index, duration, and Persistent Volume Claim (PVC) storage. This CLI configuration provides a better control over disk usage and rotation for OCS, BNG, and audit logs, with settings persisting across system restarts.

This feature allows you to configure log size, and uses a persistent volume claim (PVC) to store logs within the cnAAA environment. The `consolidated-aaa-logging-0` pod can manage log rotation and disk usage flexibly. The solution introduces configuration through the CLI in the Ops-Center. Administrators can adjust logging parameters such as log retention policies, and allocate storage according to specific requirements, log volume, and compliance needs.

Key configurable parameters: These settings configure log rotation parameters for three log categories: `ocs-log`, `bng-log`, `audit-log`, `api-log`, `engine-log`, and `app-log`.

Reports

- OCS RADIUS traffic (managed by `ocs-log` configuration):
  - `ocs_consolidated-retryandsuccess.csv`
  - `ocs_consolidated-retryandfailure.csv`
  - `pmz_consolidated-retryandsuccess.csv`
  - `pmz_consolidated-retryandfailure.csv`
- BNG BackOffRetry CoA Failure traffic (managed by `bng-log` configuration):
  - `coaBackOffFailureReport.csv`

Logs

- API (unified request and response logs for CRUD operations):
  - `consolidated-api.log`
- APP (for the Radius EP and Engine pod logs):
  - `consolidated-app.log`

- Engine (provides detailed information of each RADIUS transaction):
  - `consolidated-engine.log`
- Audit (managed by `audit-log` configuration):
  - `bng-device-audit.log`
  - `cpc-custrefdata_audit.log`
  - `cpc-audit-pb.log`
  - `cpc-audit-cc.log`
  - `cpc-pb-publish-svn-diff.log`
  - `policybuilder.log`

Total storage volume (`storage-size`): configurable for the PVC used by the `consolidated-aaa-logging` pod.

You can configure these options for BNG, OCS, and Audit logs:

- `max-index`: The maximum number of rotated log files to retain.
- `max-file-size`: The maximum size of a single log file before rotation.

You can configure below options for API, app and engine logs:

- `max-file-size`: The maximum size allowed for each individual log file (in MB). Configurable between 1–1000 MB.
- `max-history`: How long logs are retained before deletion (in hours).
- `total-cap-size`: The total storage allocated for each log type (in GB).

Note All configured settings (`max-index`, `max-history`, `total-cap-size`, `max-file-size`, and `storage-size`) persist across pod restarts, node reboots, and cnAAA upgrades, ensuring consistent log management.

Log sizing configuration

Configure log sizing through the Ops-Center provides granular control over storage volume and retention periods for various service logs to prevent disk overflow.

```
[beta/beta-cncps] pcf(config)# debug logging sizing [log-type/storage-size]
```

Note Supported log-type: `api-log`, `app-log`, `audit-log`, `bng-log`, `engine-log`, `ocs-log`.

Table 40: Service logs (API, application, engine)

Log type	Parameter	Range	Default	Unit
api-log	`max-file-size`	1 - 1000	500	MB
	`max-history`	1 - 72	48	Hours
	`total-cap-size`	1 - 1000	1	GB
engine-log	`max-file-size`	1- 1000	500	MB
	`max-history`	1 - 72	48	Hours
	`total-cap-size`	1 - 1000	1	GB
app-log	`max-file-size`	1 - 1000	500	MB
	`max-history`	1 - 72	48	Hours
	`total-cap-size`	1 - 1000	1	GB

Table 41: Infrastructure and compliance logs (Audit, BNG, OCS)

Log type	Parameter	Range	Default	Unit
audit-log	`max-file-size`	10 - 300	200	MB
	`max-index`	15 - 21	15	NA
bng-log	`max-file-size`	10 - 300	200	MB
	`max-index`	15 - 21	15	NA
ocs-log	`max-file-size`	10 - 300	200	MB
	`max-index`	15 - 21	15	NA

Table 42: Global storage

Parameter	Range	Default	Unit
storage-size	3 - 1000	3	GB

Operational behavior and verification

- Engine and application logs: Rotate every minute (`consolidated-engine.YYYY-MM-DD-Hours-Minutes.0.log.gz`).
- API, App, and Engine logs: Rotate hourly (`consolidated-api.YYYY-MM-DD-Hours.0.log.gz`).

Configure log management parameters in Ops-Center

Before you begin

- Shutdown the system to configure the `storage-size` parameter.

Procedure

Step 1 Log in to the Ops-Center and enter `config` command.

```
config
```

Step 2 Configure total storage volume to determine the overall disk space allocated for logs.

```
debug logging sizing storage-size <value_in_GB>
```



Caution Configure the `storage-size` parameter only when the system is in a shutdown state. Once configured, the size cannot be reduced. It can only be increased. The default and minimum value is 3 GB.

Step 3 Configure OCS RADIUS traffic log parameters to control log rotation.

- Configure the `max-index` value to specify the maximum number of rotated log files to retain.

  ```
  debug logging sizing ocs-log max-index <value>
  ```

- Configure the `max-file-size` value to specify the maximum size of a single log file before rotation.

  ```
  debug logging sizing ocs-log max-file-size <value_in_MB>
  ```

Step 4 Configure BNG RADIUS traffic log parameters to control log rotation.

- Configure the `max-index` value to specify the maximum number of rotated log files to retain.

  ```
  debug logging sizing bng-log max-index <value>
  ```

- Configure `max-file-size` value to specify the maximum size of a single log file before rotation.

  ```
  debug logging sizing bng-log max-file-size <value_in_MB>
  ```

Step 5 Configure `audit-log` parameters to control log rotation for audit-related logs.

- Configure the `max-index` value to specify the maximum number of rotated log files to retain.

  ```
  debug logging sizing audit-log max-index <value>
  ```

- Configure the `max-file-size` value to specify the maximum size of a single log file before rotation.

  ```
  debug logging sizing audit-log max-file-size <value_in_MB>
  ```

Step 6 Configure `api-log` parameters to control log rotation for api-related logs.

- Configure the `max-file-size` value to specify the maximum size of the log file before rotation.

  ```
  debug logging sizing api-log max-file-size <value_in_MB>
  ```

- Configure the `maxhistory`value to specify the maximum retention period for logs.

  ```
  debug logging sizing api-log max-history <value_in_hours>
  ```

- Configure the `total-cap-size` value to specify the total storage cap for API logs.

  ```
  debug logging sizing api-log total-cap-size <value_in_GB>
  ```

Step 7 Configure `app-log` parameters to control log rotation for app-related logs.

- Configure the `max-file-size` value to specify the maximum file size before rotation .

```
debug logging sizing app-log max-file-size <value_in_MB>
```

- Configure the `max-history` value to specify the maximum retention period for logs.

```
debug logging sizing app-log max-history <value_in_hours>
```

- Configure the `total-cap-size` value to specify the total storage cap for application logs.

```
debug logging sizing app-log total-cap-size <value_in_GB>
```

Step 8 Configure `engine-log` parameters to control log rotation for app-related logs.

- Configure the `max-file-size` value to specify the maximum file size before rotation.

```
debug logging sizing engine-log max-file-size <value_in_MB>
```

- Configure the `max-history` value to specify the maximum retention period for logs.

```
debug logging sizing engine-log max-history <value_in_hours>
```

- Configure the `total-cap-size` value to specify the total storage cap for application logs.

```
debug logging sizing engine-log total-cap-size <value_in_GB>
```

Step 9 `commit` the changes and verify the current configurations by using the `show running-config` command.

```
show running-config debug logging sizing
```

Caution Ensure that the combined `total-cap-size` of the engine, API, and app logs does not exceed the configured `storage-size`.

Note

- The recommended `max-history` for API, app, and engine logs is **48 hours**. The recommended `max-file-size` is **500 MB** for all log types.
- The system may store data beyond the configured max-history period if the total-cap-size for that log type is not reached.

Example:

```
debug logging sizing storage-size 200
debug logging sizing ocs-log max-index 20
debug logging sizing ocs-log max-file-size 250
debug logging sizing bng-log max-index 20
debug logging sizing bng-log max-file-size 250
debug logging sizing audit-log max-index 20
debug logging sizing audit-log max-file-size 250
```

```
debug logging sizing api-log max-file-size 500
debug logging sizing api-log max-history 48
debug logging sizing api-log total-cap-size 1
debug logging sizing app-log max-file-size 500
debug logging sizing app-log max-history 48
debug logging sizing app-log total-cap-size 1
debug logging sizing engine-log max-file-size 500
debug logging sizing engine-log max-history 48
debug logging sizing engine-log total-cap-size 1
```

Log forwarding to external servers

Feature History

Feature Name	Release Information	Description
Log forwarding to external servers	2026.01.0	This feature forwards pod and application logs to external Syslog, Splunk, and Fluentd servers by using CEE and CPC Ops-Center configurations. It supports IPv4 and IPv6 addresses to provide centralized log management and resolve multi-line rendering issues. It forwards audit logs to an external syslog server.

The Log Forwarding feature allows CPC to export pod logs (application and MongoDB logs) to external Syslog, Splunk, and Fluentd servers. This feature supports both IPv4 and IPv6 addressing to accommodate diverse customer network environments. It also forwards audit log to the external syslog server.

The system uses two management interfaces for log forwarding:

- CEE Ops-Center forwards general logs to Syslog, Splunk, and Fluentd.
- The CPC Ops-Center forwards application logs to Splunk, which helps resolve multi-line logging issues.

Configure log forwarding in the CEE and CPC Ops-Centers

Before you begin

- Configure reachability alerts before setting up the external server. This allows the system to monitoring the connection immediately when it is activated.

```
cee(config)# alerts rules group fluentbit.rules rule fluent-proxy-output-retries-failed
 expression "rate(fluentbit_output_retries_failed_total{pod=~\"fluent-proxy.*\"}[5m])
> 0" duration 5m severity major type "Communications Alarm" annotation description value
 "Fluent-proxy {{ $labels.namespace }}/{{ $labels.pod }} output retries failed for
target: {{ $labels.name }}"
```

- Log forwarding is disabled by default. This functionality is enabled only after the configuration is commited.

- To verify connectivity, check that the external server is reachable using the configured IPv4 or IPv6 address.

Procedure

Step 1 Log in to Ops-Center and enter the configuration mode.

```
config
```

Step 2 The CEE Ops-Center supports forwarding to three types of external collectors. Configure the collector as per the requirement.

a. Configure the Syslog configuration details.

```
cee(config)# logging syslog host 192.0.2.10 syslog-message-key MESSAGE workers 4 syslog-hostname-key
 _HOSTNAME syslog-appname-key CONTAINER_NAME
```

b. Configure the Splunk configuration details.

```
cee(config)# logging splunk host 192.0.2.10 port 8088 auth-token <your_token>
disable-tls-verification true
```

c. Configure the Fluentd configuration details.

```
cee(config)# logging fluent host 192.0.2.10 port 24224 protocol forward enable
```

Step 3 *commit* the changes and verify the status.

- Verify the CEE Syslog

```
show running-config logging syslog
```

- Verify the CEE Fluentd

```
show running-config logging fluent
```

- Verify the CEE Splunk

```
show running-config logging splunk
```

- Verify the CEE alerts

```
show running-config alerts rules group fluentbit.rules
```

Step 4 Use the CPC Ops-Center CLI to resolve multi-line logging issues and ensure application logs forward correctly to Splunk.

- Configure the Splunk details.

```
pcf(config)# debug splunk hec-url https://198.51.100.20:8088 hec-token <your_token>
```

Step 5 `commit` the changes and verify the status.

- Verify the CPC Splunk

```
show running-config debug splunk
```

Note If application logs are forwarded using CPC Ops-Center, exclude those logs from CEE Ops-Center to prevent duplication or redundancy in logging. For more information refer: https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smi/cee-config-guide/ucc-5g-smi-cee-config-and-admin-guide/m_cisco-smi-cee-logging.html

Sample configuration:

```
show running-config logging syslog
Fri Oct  10 12:05:26.268 UTC+00:00
logging syslog host <Repo_Server_IP>
logging syslog syslog-hostname-key _HOSTNAME
logging syslog syslog-message-key MESSAGE
logging syslog syslog-appname-key CONTAINER_NAME
logging syslog workers 4
[alpha/alpha] cee#

[beta/beta-cncps-cee] cee# show running-config logging fluent
Fri Jan  23 04:21:43.429 UTC+00:00
logging fluent enable
logging fluent host <Repo_Server_IP>
logging fluent port 24224
logging fluent protocol forward

[beta/beta-cncps-cee] cee# show running-config logging splunk
Fri Jan  23 04:21:38.190 UTC+00:00
logging splunk host      <Repo_Server_IP>
logging splunk port      8088
logging splunk auth-token b847e79b-d45a-450e-8496-5f072dd9d205
logging splunk disable-tls-verification true

[alpha/alpha] cee# show running-config alerts rules group fluentbit.rules
Wed Dec  24 04:31:50.881 UTC+00:00
alerts rules group fluentbit.rules
 rule fluent-proxy-output-retries-failed
  expression "rate(fluentbit_output_retries_failed_total{pod=~\"fluent-proxy.*\"}[5m]) >
0"
  duration   5m
  severity   major
  type       "Communications Alarm"
  annotation description
   value "Fluent-proxy {{ $labels.namespace }}/{{ $labels.pod }} output retries failed for
 target: {{ $labels.name }}"
  exit
exit
exit

[unknown] pcf# show running-config debug splunk
Fri Jan 23 13:37:44.728 UTC+00:00
debug splunk hec-url https://10.81.68.212:8088
debug splunk hec-token 4e86ba95-615f-48dd-9711-a9a9d0b60487
```

Known limitations

- The CPC Ops-Center configuration does not currently support alerts for external server reachability.

- Enabling log forwarding through CEE Ops-Center deploys fluent-bit worker pod on each node and one fluent-bit proxy pod on one of the master node. This increases CPU and memory utilization.

Grafana database monitoring dashboard

The Grafana database monitoring dashboard now provides improved visibility into database record counts. Previously, the dashboard aggregated all database records into a single panel, which made it difficult to distinguish between different data types.

In this release, the dashboard features segregated panels to allow for more granular monitoring:

- **SPR records panel**: Displays the record count specifically for SPR data.
- **CRD records panel**: Displays the record count for CRD, which includes the admin and admin-db databases.



Note The system categorizes CRD data under the `admin` and `admin-db` labels within the monitoring system.

Grafana dashboard files

Use this reference to identify each supplied dashboard file and its monitoring area.

The JSON files provide separate cnAAA monitoring views in Grafana.

Prerequisites

Before you import the dashboards, ensure that these JSON files are available:

- `cnAAA_Radius_TPS-KPIs.json`: RADIUS TPS monitoring.
- `cnAAA_Pod_Level_CPU_Memory_Utilization_KPIs.json`: Pod-level CPU and memory utilization.
- `cnAAA_Node_Level_CPU_Memory_Utilization_KPIs.json`: Node-level CPU and memory utilization.
- `cnAAA_Resource_Utilization_KPIs.json`: CPU and memory monitoring.
- `cnAAA_Soap_API_KPIs.json`: SOAP API monitoring.
- `cnAAA_Soap-Proxy_CPC_Migration_KPIs.json`: SOAP Proxy and CPC migration monitoring.

For information about file availability or for assistance importing dashboards, contact the Cisco cnAAA engineering team at cnaaa-ft-svi-team@cisco.com.

Audit and logging enhancements

Overview

Feature Name	Release Information	Description
Audit and logging enhancement	2026.03.0	The audit and logging enhancements tracks login attempts, configuration changes, and administrative actions across the Policy Builder, Control Center, and BNG subsystems. Events are automatically forwarded to the Centralized Log Management System (CLMS) to support the enhancement supports security auditing, troubleshooting, and regulatory compliance.

The CPC platform provides audit logging across the Policy Builder, Control Center, and BNG audit subsystems by capturing authentication failure details, exact configuration diffs for Policy Builder commits, and user attributed actions of BNG audit logs. All audit logs are forwarded to the CLMS as a permanent security and compliance record.

Audit and logging enhancement provides these capabilities:

- Captures all failed login events for the Policy Builder and Control Center.
- Records the exact Subversion (SVN) diff content for every configuration commit performed through the Policy Builder GUI.
- Appends the user identity to BNG audit logs to link each administrative actions to the initiating user.
- Logs subscriber-level administrative actions, including adding, updating, and deleting subscribers.
- Automates the forwarding of audit logs to the CLMS for forensic and compliance review.

Standardized log schema

Every audit log entry follows a standardized format to ensure consistency across subsystems. Each entry must include these five parameters in a single line:

Table 43: Standardized log schema

Parameter	Description
Source IP	The originating client IP address of the user.
Username	The identity of the user performing the action.
Timestamp	The precise time of the event (ISO 8601 format).
Status	The outcome of the operation (e.g., SUCCESS, FAILURE).

Parameter	Description
Activity Details	A summary of the configuration or administrative action, including diff context or subscriber details.

Policy Builder Configuration Tracking

The `audit log` provides traceability for configuration commits by showing the transition from the previous configuration to the current state.

Successful logs:

```
Source IP:<SourceIP>, Username: <UserName>, Timestamp:<timestamp>, Session ID: <sessionId>,
 Status: SUCCESS, Activity: Login attempt, Details: User logged-in successfully to Policy
Builder
```

Failure logs:

```
Source IP: <source IP>, Username: <username>, Timestamp: <timestamp>, Status: <status>,
Activity: Login attempt, Details: Authentication failed. Enter a valid username and password,
 or check if your password has expired.
```

PB publish:

```
SVN DIFF User: <svn diff user>, Source IP: <source IP>, Session ID: <session ID>, Activity:
 PB Publish, commit version: <commit version>, previous commit version: <previous commit
version>, Previous Config: "<config>"; , Current Config Change: "<config change>"; Commit
Message: User: <user> published with Commit Message: <commit message>
```

Logout:

```
Logout, User: <user>, Source IP: <source IP>, Session ID: <session ID>, User logged-out
successfully from Policy Builder
```

- Policy builder logs will be present `cpc-audit-pb.log`.
- Control center logs will be present in `cpc-audit-cc.log`.
- BNG logs will be present in `bng-device-audit.log`.

Note

1. After you perform the first publish operation in Policy Builder, log flooding may occur. This happens because the system imports existing services. This is expected behavior.
2. Publish operation contains the metadata which includes `commit version`, `previous commit version`, `Previous Config`, `Current Config Change`, and `Commit Message` instead of `Details`.

Control Center Activity Logging

The Control Center logs successful and failed login attempts, as well as subscriber-level administrative actions.

Successful Login:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Login attempt,
Details: User logged-in successfully to Control Center with sessionId: <session ID>
```

Failed Login:

```
Source IP: <source IP>, Username: <username>, Status: FAILED, Activity: Login attempt,
Details: Invalid username or password - Bad credentials
```

Logout:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Logout attempt,
Details: Logout successful
```

Subscriber and SSID Management

Add Subscriber:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Add Subscriber,
Details: Created subscriber with credential Id: demo
```

Update Subscriber:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Update Subscriber,
 Details: Updated subscriber with credential Id: demo, services: 1, notifications: 1,
subAccounts: 1
```

Add/Delete SSID:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Add SSID, Details:
 Added SSID for subscriber credential Id: demo, SSID Key: abc
```

Delete Subscriber:

```
Source IP: <source IP>, Username: <username>, Status: SUCCESS, Activity: Delete Subscriber,
 Details: Deleted subscriber with credential Id: demo
```

Configure CLMS log forwarding

These log files content must be forwarded to the external CLMS server: cpc-audit-pb.log, cpc-pb-publish-svn-diff.log, bng-device-audit.log, and cpc-audit-cc.log.

The existing CEE Ops-Center CLI is used to forward logs to the CLMS server.

Procedure

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

Example:

```
ssh -p <port_number> admin@<cee_ops_center_ip>
```

Step 2 Enter configuration mode.

```
config
```

Step 3 Add this configuration to forward logs to CLMS server.

```
logging fluent enable
logging fluent host <CLMSServerIP>
logging fluent port 24224
logging fluent protocol forward
```

Step 4 `commit` the changes.

Verify audit logging

Use this procedure to confirm that the system successfully records and forwards audit logs.

Procedure

Step 1 Log in to the Policy Builder and Control Center using an incorrect password. Then verify that an entry is recorded in `cpc-audit-pb.log` and `cpc-audit-cc.log`:

```
:~$ kubectl logs -n <namespace> consolidated-aaa-logging-0 | grep -i "Login Attempt"
```

Step 2 Verify that the Fluent Bit configurations are present.

```
show running-config logging fluent
```

Step 3 Check the status of the Fluent Bit pod to confirm that log forwarding functions correctly.

```
kubectl get pods -n <cee_namespace> | grep fluent-bit
```

Log noise reduction

Log noise reduction reduces unnecessary logging in a CPC deployment. It prevents duplicate scheduler execution outside the cnAAA Engine and suppresses high-volume warmup-subscriber RADIUS messages.

This enhancement resolves these issues:

- Prevent duplicate scheduler execution in the CRD, Control Center, and Policy Builder components.
- Retain the intended scheduler execution in the cnAAA engine.
- Suppress high-volume logs that are generated by RADIUS call flows for warmup subscribers.

Affected components and behavior

This table summarizes the behavior changes for the affected components and processes.

Table 44: Log noise reduction impacts

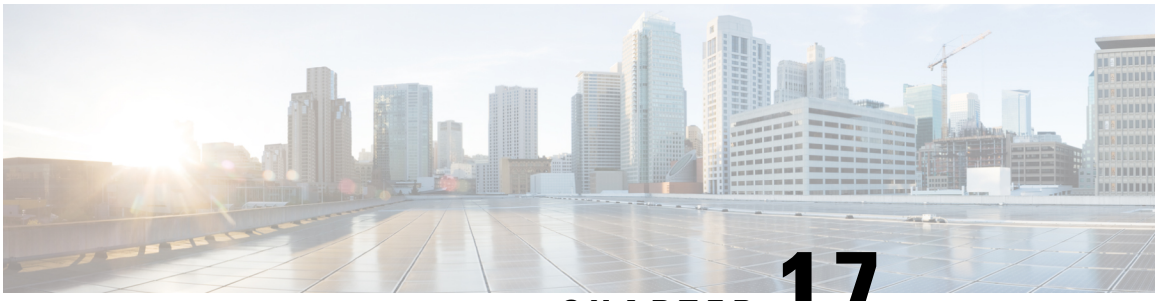
Component	Behavior change
cnAAA Engine	Executes the intended schedulers.
CRD, Control Center, and Policy Builder	Do not execute duplicate schedulers, such as SVNMonitoringKPIScheduler or DiagnosticController, that are identified by their feature.
Warmup-subscriber processing	Suppresses unnecessary messages from the RADIUS call flow.

The enhancement addresses log noise from schedulers that were copied into multiple CPC components. The identified schedulers are intended to run in the cnAAA Engine.

Warmup-subscriber processing uses a RADIUS call flow. Repeated cnAAA Engine restarts can cause the related messages to accumulate and make log analysis difficult.



Note No configuration by the user is necessary to enable this enhancement. When the new software build is installed, the CPC software automatically applies the code and logger changes.



CHAPTER 17

Policy Builder Overview

- Overview, on page 321
- Reference Data, on page 322
- Services, on page 323
- Policies, on page 332
- Add a System, on page 333
- Access the Policy Builder, on page 334
- Add and update the services, validation, on page 335
- Create a new RADIUS Service Template, on page 337
- Policy Enforcement Points, on page 338
- Advantages, on page 346
- UI changes, on page 348
- Manage OCS and BNG configurations with single API, on page 350
- SOAP API engine rate limiting, on page 362

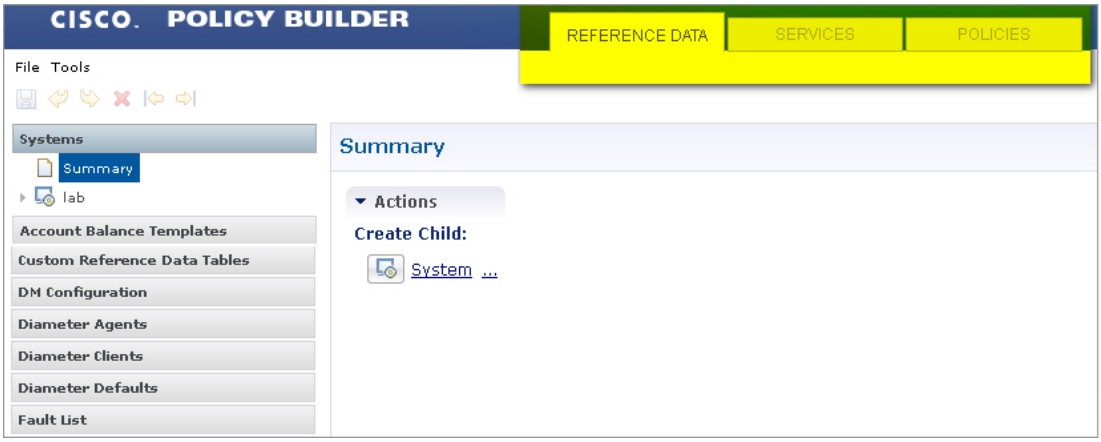
Overview

Converged Policy and Charging (CPC) provides a framework for building rules that can be used to enforce business logic against policy enforcement points such as network routers and packet data gateways. For example, a prepaid customer (one who pays as they go) might be denied service or prompted to top-up when their quota has expired, whereas a postpaid customer (one who has an ongoing billing relationship with the service provider) might only have their service downgraded or be automatically billed for additional data when their particular quota has expired.

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. The following 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.

Figure 24: Cisco Policy Guilder GUI



Reference Data

The Reference Data tab of the PB GUI provides access for configuring various aspects of the system in order to make the system ready for operation. Reference Data are used to not only configure the system, but are also used to provide settings and parameters that are referenced by policy rules across various services; for example, Account Balances and Notifications are configured as Reference Data but are then referenced and reused by multiple services as needed. Details of the various Reference Data configuration options are described in more detail in other chapters of this guide.

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

- Systems, cluster, and instance data
- Jdbc query string definitions
- Balance and quota definitions
- RADIUS agents, clients, and defaults information
- 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
- Tariff switch times
- Fault list

Services

Feature Summary and Revision History

Summary Data

Table 45: Summary Data

Applicable Product(s) or Functional Area	CPC
Applicable Platform(s)	SMI
Default Setting	Enabled – Always-on
Related Documentation	Not Applicable

Feature History

Table 46: Feature History

Feature Details	Release
First introduced.	2025.01.0

Important Due to non-backward compatible changes in cnAAA operations center configuration model, a direct cnAAA upgrade is not possible. You must perform a fresh cnAAA installation after un-deploying the previous installation and clearing out the cnAAA configmaps from CNEE.

Feature Description

A service dictates the capabilities that are assigned to a subscriber (in USuM). An administrator assigns a service to a user through the service configurations. Depending on the service provider's requirements, cnAAA lets you flexibly map the service configuration with the policies.

For instance, a user with the GOLD account might get a high upload/download speed in comparison to a BRONZE user.

In a tier-based classification, if the quota is "y" then the users from the first tier are redirected to a portal and users belonging the second tier would only experience a downgrade in the speed.

Service

A service is effectively a "code" to label the service and a collection of Service Options which contain the definition of what a service is. Multiple services can be assigned to a single subscriber. If multiple services are assigned to a subscriber, the service options are combined between all assigned services.

Adding a Service

Before adding a service, ensure that you have created the corresponding Use Case template for the service that you intend to add.

Use the following steps to add a service through Policy Builder.

1. Log in to Policy Builder.
2. Click the **Use Case Templates** from the left pane and select the template that you have created.
3. In the right pane, click **Add** to include a new service.
4. In the **Select Service Configuration** dialog box, click the appropriate entry to view the associated services.
5. Select the service and click **OK**. The selected service is added as a new service.
6. In the left pane, choose **Services** > **Service Options** to view the options.
7. Expand the service that you have created and select the child.



Note The service name resembles the name that you specified for the use case template.

8. In the **Service Option** pane, click the service under **Service Configurations** and specify the parameters referring to the relevant configuration.

Service configuration

cnAAA uses the low-level configuration objects to drive a feature in the system. You can configure the Service Configuration objects from the **Service** > **Service Option** > **Use Case Template**.

Types of service configurations:

- PriorityConfiguration: Only one configuration is allowed to be active at a time. If multiples priority configurations are added, the configuration of the highest priority is used. These are used in cases where only a single value makes sense. For example, when sending an Accept message, only one template is required. Objects of this type always have a priority field. If multiple priority configurations are added, the highest priority object is used. For example, AccessAcceptConfiguration and RegisterMacAddress.
- GroupConfiguration (most common): Only 1 configuration per 'Group Name' is allowed to be active. If multiple configurations are added, the highest priority per Group Name is used. These configurations are used in cases where a configuration only makes sense for a single "group" (key). For example, to control the upload/download speed based on the network type (cell, Wi-Fi, and so on). A service configuration to control network speed with a group set for cell/Wi-Fi would allow multiple service configurations to be added. These objects always have a group field and a priority field. For each unique group value, the highest priority is used. For example, ServiceConfiguration, All RADIUS Configurations, and OneTimeUsageCharge.
- ServiceConfiguration: Multiple configurations are allowed. If multiple configurations are added, all are used. For example, AutoChargeUpAccounts, AutoProvisionQuota, and BalanceRateConfiguration.

Note The Modify feature in PB for Use Case Options/Service Options can override the values conditionally.

Use Case Templates

Use case templates are the essential elements of the CPC architecture. The values that you define in the templates allow you to design and configure one or more services once and reuse them.

Only advanced users such as administrators are authorized to create a use case template.

On a higher-level, the use case template lets you:

- Define the Service Configuration objects to be set by a Service Option.
- Provide default values and hide values which the use case must not configure.
- Optionally, contains Initiators (Conditions) which define when the template is active.
- Makes Service Option and Service creation easier. For example, a use case template setup to create different upload or download speeds includes a DefaultBearer QoS Service Configuration object. The user creating a use case template can set default and hide the values for ARP and other values that are not directly related to upload or download speed. This allows the creation of the Service Option to be much simpler.
- A copy of the Use Case Options is created while copying a use case template.

Configure the Use Case Template

This section describes how to configure the use case template.

Use the following steps to configure the use case template through Policy Builder.

1. Log in to Policy Builder.
2. Select the **Services** tab, and from the left pane click **Use Case Templates** to create a new service.
3. On the left pane, click **Summary** to open the **Summary** pane.
4. Under **Actions**, click **Use Case Template**.
5. In the **Use Case Template** pane, specify the name for the template.
6. Click the **Actions** tab and select **Add**.
7. In the **Select Service Configuration** dialog box, select the service and click **OK**. The **Use Case template** with the specified name is created.
8. In the left pane, click **Services > Service Options** to view the options. The newly created service appears in the **Service Options**.
9. Select the service that you have created.
10. Under **Service Configurations**, click **Add** to open the **Select Service Configuration** dialog box.

11. Under **Service Configurations**, select the service, then click **OK**.

Generic service configuration

This section describes the parameters for the Generic Service Configuration service configuration object.

Table 47: Generic Service Configuration Parameters

Parameters	Description
Priority	Denotes the priority of the message for processing. The higher the number, the higher the priority. Default for most settings: 0
Group Name	Specifies a group name. Only 1 per Group Name is allowed to be active. If multiple configurations are added highest priority per Group Name is used.
Code	Specifies a code for the AVP.
Value	Specifies a value for the AVP.
String Value	Specifies the string value.
Int Value	Indicates the integer value.
Long Value	Indicates the long value.
Boolean Value	Specifies the boolean value.
String Value to Override	Indicates whether overriding is required. For virtual services, if the value of “String Value” field matches exactly with the value of “String Value To Override”, then the value of “String Value” is over written with the “New String Value".
New String Value	The new string value that is used to overwrite the “String Value” if the value of “String Value” field matches exactly with the value of “String Value To Override”.
Precedence	Defines the second-level priority when the highest priority matches among the multiple generic service configurations.

Common Parameters

These parameters are common between many service configuration objects.

Table 48: Common Service Configuration Object Parameters

Parameter	Description
Apn Agg Max Bit Rate DL	Defines the total bandwidth usage for the downlink direction of non-GBR QCIs at the APN.
Apn Agg Max Bit Rate UL	Defines the total bandwidth usage for the uplink direction of non-GBR QCIs at the APN.
Arp	AllocationRetentionPriority • Priority Level – Priority-Level AVP value. • Preemption Capability – Preemption-Capability AVP value. • Preemption Vulnerability – Preemption-Vulnerability AVP value.
Balance Code	Indicates with which balance the quota is associated. You can subscribe to multiple balances, but the monitoring key is associated with one balance.
Dosage	How much quota to initially give the client (in bytes). Default: 0
Dual Stack Session	Set to enable or disable the parameter. Default: disabled
Enable Resource Allocation Notification	Can be set to enabled or disabled. Default: disabled
Encoding Format	Can be set to true or false. If the Monitoring Key parameter is numeric, set this parameter to true. Default: false
Event Trigger	Used primarily to notify the starting and stopping of applications or to report usage. It is not used to rerequest rules.
Flow Status	Defines whether the service data flow is enabled or disabled.
Framed I P Type	Can be set to one of the following options: • ANY_ONE • BOTH • IPv4_ADDRESS • IPv6_ADDRESS Default: ANY_ONE
Guaranteed Bit Rate DL	Defines the guaranteed bit rate allowed for the downlink direction.

Parameter	Description
Guaranteed Bit Rate UL	Defines the guaranteed bit rate allowed for the uplink direction.
List of Input Column Avp Pairs (List)	Defines the mapping between the AVP Names and the key columns defined in the selected STG. These AVPs are used as inputs while evaluating the CRD table in STG. • Avp Name – The name of the RADIUS AVP that is used as input for CRD table evaluation. For example: Flow-Number, Media-Component-Number, and so on. • Column – The key column in STG that corresponds to the specified AVP.
List Of Output Column Avp Pairs (List)	Defines the mapping between the AVP Names and the output columns defined in the selected STG. These mappings indicate how the output columns values are mapped to AVPs after the CRD is evaluated. • Avp Name – The name of the RADIUS AVP to which the value of the output column is mapped while setting the charging parameters on the dynamic rule (for the Dedicated Bearer). For example: Rating-Group Service-Identifier. • Column – The output column defined in the selected STG.
Max Req Bandwidth DL	Defines the maximum bit rate allowed for the downlink direction.
Max Req Bandwidth UL	Defines the maximum bit rate allowed for the uplink direction.
Monitoring Key	Identifies a usage monitoring control instance. You can specify any value.
Monitoring Level	Can be set to one of the following values: • SESSION_LEVEL (0) • PCC_RULE_LEVEL (1) • ADC_RULE_LEVEL (2)
Mute Notification	Indicates whether notifications for application starts and stops are muted for ADC Rule by the TDF.
New String Value	The new string value that is used to overwrite the “String Value” if the value of “String Value” field matches exactly with the value of “String Value To Override”.

Parameter	Description
Online	Defines whether the online charging interface from cnAAA for the associated PCC rule is enabled. The default charging method provided by cnAAA takes precedence over any preconfigured default charging method at cnAAA. • Enable: Indicates that the online charging interface for the associated PCC rule is enabled. • Disable: Indicates that the online charging interface for the associated PCC rule is disabled.
Offline	Defines whether the offline charging interface from cnAAA for the associated PCC rule is enabled. The default charging method provided by cnAAA takes precedence over any preconfigured default charging method at cnAAA. • Enable: Indicates that the offline charging interface for the associated PCC rule is enabled. • Disable: Indicates that the offline charging interface for the associated PCC rule is disabled.
Precedence	Defines the second-level priority when the highest priority matches among the multiple generic service configurations.
Preemption Capability	When provided within the QoS-Information AVP, the AVP defines whether a service data flow can get resources that were already assigned to another service data flow that has a lower priority level. If it is provided within the Default-EPS-Bearer-QoS AVP, the AVP defines whether the Default Bearer can get resources that were already assigned to another bearer with a lower priority level. • 0: Indicates that the service data flow or bearer is allowed to get resources that were already assigned to another service data flow or bearer with a lower priority level. • 1: Indicates that the service data flow or bearer is not allowed to get resources that were already assigned to another service data flow or bearer with a lower priority level. This is the default value applicable if this AVP is not supplied.

Parameter	Description
Preemption Vulnerability	When provided within the QoS-Information AVP, the AVP defines whether a service data flow can lose the resources assigned to it in order to admit a service data flow that has a higher priority level. If it is provided within the Default-EPS-Bearer-QoS AVP, the AVP defines whether the Default Bearer can lose the resources assigned to it in order to admit a pre-emption capable bearer with a higher priority level. • 0: Indicates that the resources assigned to the service data flow or bearer can be pre-empted and allocated to a service data flow or bearer with a higher priority level. This is the default value applicable if this AVP is not supplied. • 1: Indicates that the resources assigned to the service data flow or bearer cannot be pre-empted and allocated to a service data flow or bearer with a higher priority level.
Priority	The priority of the message for processing. The higher the number, the higher the priority. Default for most settings: 0
Priority Levels	Used to decide whether a bearer establishment or modification request can be accepted, or rejected due to resource limitations (typically used for admission control of GBR traffic). The AVP can also be used to decide which existing bearers to pre-empt during resource limitations. The priority level defines the relative importance of a resource request. Values 1–15 are defined, with value 1 as the highest level of priority. • Values: 1–8 – Assigned for services that are authorized to receive Prioritized treatment within an operator domain. • Values: 9–15 – Assigned to resources that are authorized by the Home network and thus applicable when a UE is roaming.
Provision Default Bearer QoS	Must be bound to the appropriate column in the STG. The data contained in the STG column is of type True/False. If the value is True, the Default Bearer QoS information from the session is applied to the rule, while QoS information derived from the prior parameters in this STG is ignored.

Parameter	Description
Qci	The Quality of Service (QoS) Class Identifier. The QoS class identifier identifies a set of IP-CAN specific QoS parameters that define QoS, excluding the applicable bitrates and ARP. It is applicable both for uplink and downlink direction. The QCI values 0, 10–255 are divided for usage as follows: • 0: Reserved • 10-127: Reserved • 128-254: Operator specific • 255: Reserved
Rating Group	The charging key for the PCC rule used for rating purposes.
Realm	The destination realm where the message is sent from cnAAA.
Redirect Address	Indicates the target for redirected application traffic.
Redirect Address Type	Defines the address type of the address given in the Redirect-Server-Address AVP. Default: IPV4_ADDRESS
Redirect Server Address	Indicates the target for redirected application traffic.
Redirect Support	This value indicates that Redirection is enabled for a detected application's traffic.
Retry Profile	Indicates the Rule Retry Profile to be used. When cnAAA receives a Charging-Rule-Report indicating failure to install or to activate one or more rules, it evaluates the failed rules and takes further action.
Rule Group	Used to classify rules at cnAAA to change set of predefined rules based on policy. This parameter is optional.
Rule Name	A partial name configured in Policy Builder (as derived using AF-Application-Identifier and Media-Type values from the Custom dynamic rule name table in Gx Client). Default: AF

Parameter	Description
Scheduled Hour	Can be set to one of the following values: • Default: Turns off the Hour Boundary RAR enhancement feature for look-ahead rules installation at hour boundary. This causes rules to be installed at hour boundary as applicable. • CurrentHour: Rule activation time will be current time, deactivation time will be the next hour. • NextHour: Rule activation time will be the next hour, and deactivation time will be next-next hour.
Search Column	Must be bound to the Key column in the STG. The data contained in the STG column is of type Text.
Search Group	A constant value that cnAAA uses to search within the Search Table Group indicated by the Search Table parameter.
Search Table	The name of the table from which to perform a lookup.
String Value to Override	Indicates whether overriding is required. For virtual services, if the value of “String Value” field matches exactly with the value of “String Value To Override”, then the value of “String Value” is over written with the “New String Value".
Tdf Application Identifier	References the application detection filter (for example, its value may represent an application such as a list of URLs) to which the PCC rule for application detection and control in cnAAA applies.
ToD Schedule	Identifies the schedule for rule activation and deactivation.

Policies

While the Services tab, through Use Case Templates and Service Options, makes it easy to create reusable and extensible services, the Policies tab allows direct access to the underlying policy engine. The Policies tab holds the cnAAA core system Blueprint, which is composed of various Extension Points that break the policy engine flow into sections that occur within the execution of the policy. For example, the point in the policy flow where a Gx connection is received, parsed, and processed before the point in the policy flow where the related subscriber data is evaluated.

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.

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.

Summary of Policy Tab Capabilities

- 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

- Building custom policies requires a deep understanding of the call flow and underlying cnAAA 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

Add a System

This section describes how to add a system.

After installation, use this procedure to set up your Policy Builder by using an example populated with default data. You can change anything that does not apply to your deployment.

1. Click the **Reference Data** tab, and then click the **Systems** node to display the **Systems** tree.
2. Click **System...** under **Create Child:** to open the **System** pane on the right side.
3. Fill in the **Name** field, and provide a description of this system. Enter the rest of the parameters based on your network requirements.

Table 49: System Parameters

Parameter	Description
Name	The name of the cnAAA system.
Description	Describes the system using which you can uniquely identify the system.

Parameter	Description
Session Expiration Hours	An event occurs whenever a session is updated, which in turn increments the session expiry duration. If no session update event occurs in the specified session expiration duration (combination of **Session Expiration Hours** and **Session Expiration Minutes**), then the session will be removed. **Note** The combined value of **Session Expiration Hours** multiplied by 60 plus **Session Expiration Minutes** should not exceed 35,400 minutes. Default value is 8.
Session Expiration Minutes	An event occurs whenever a session is updated, which in turn increments the session expiry duration. If no session update event occurs in the specified session expiration duration (combination of **Session Expiration Hours** and **Session Expiration Minutes**), then the session will be removed. **Note** The combined value of **Session Expiration Hours** multiplied by 60 plus **Session Expiration Minutes** should not exceed 35,400 minutes. Default value is 0.

Access the Policy Builder

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

The Policy Builder provides a PAM based and SVN based authentication mechanism to support the authentication of Linux user credentials. The `disablePamAuthentication` flag is used to enable or disable user login and to perform PAM based authentication.

The following tables describes the user roles and credentials supported:

Table 50: Supported User Roles and Credentials

Linux access	SVN access	User access to Policy Builder	User Roles	Authentication Mechanism
Read/Write	Not an SVN user	Yes	Read only	PAM (Linux Systems) (set disablePamAuthentication = false)
Read only	Not an SVN user	Yes	Read only	PAM (Linux Systems) (set disablePamAuthentication = false)
Read/Write	Read/Write	Yes	Admin	PAM (Linux Systems) (set disablePamAuthentication = false)

Linux access	SVN access	User access to Policy Builder	User Roles	Authentication Mechanism
Read/Write	Read only	Yes	Read only	PAM (Linux Systems) (set disablePamAuthentication = false)
Read only	Read/Write	Yes	Admin	PAM (Linux Systems) (set disablePamAuthentication = false)
Read only	Read only	Yes	Read only	PAM (Linux Systems) (set disablePamAuthentication = false)
Not a Linux user	Read only	Yes	Read only	SVN (set disablePamAuthentication = true)
Not a Linux user	Read/Write	Yes	Admin	SVN (set disablePamAuthentication = true)
Not a Linux user	Not an SVN user	No	Invalid username or password error	PAM/SVN

cnAAA 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.

URL to access interface

Follow these steps to get to the PB URL:

1. Execute this command :

   ```
   kubectl get httproutes -n <pcf namespace> | grep pb
   ```

   Execute this command get this output:

   ```
   policy-builder-httproutes-pcf-beta-cncps-pcf-engine-app-production-rjio nginx
   pb.<namespace>-pcf-engine-app-production-rjio.<ip>.nip.io <ip>
   ```

2. Build the PB URL:

   ```
   https://<httproutes URL given in above command output>/pb
   ```

   ```
   https://pb.<namespace>-pcf-engine-app-production-rjio.<ip>.nip.io/pb
   ```

Add and update the services, validation

Policy Builder in the Cloud Native Authentication, Authorization, and Accounting (cnAAA) is a web-based interface that allows service providers to manage and configure policy settings through HTTPS. This GUI facilitates the configuration of various components such as Radius Configuration, Radius Service Template, Policy Enforcement Points, Subscriber Data Sources, and Tariff Times. This helps in managing and updating services without disrupting the active subscriber sessions and supports multi-user collaboration.

RADIUS

Remote Authentication Dial-In User Service (RADIUS), is a networking protocol that provides centralized management for Authentication, Authorization, and Accounting (AAA) for users connecting to and using a network service. This protocol is essential for ensuring secure and efficient access control within network environments.

In a converged network environment, configuring RADIUS settings involves managing policies across different network elements, a task that can be complex and time-consuming. Network administrators often encounter challenges in ensuring consistent policy configurations and maintenance across various systems and devices. RADIUS simplifies policy management by providing a central framework, improving efficiency and security in network access.

Radius Configuration

Click **RADIUS Configuration** in the right pane to add the configuration in the system.

The following parameters can be configured under RADIUS Configuration:

Table 51: RADIUS Configuration Parameters

Parameter	Description
Accounting Port	Port used for incoming radius accounting.
Authorization Port	Port used for incoming radius authorization.
Coa Port	Port used for Change of Authority between cnAAA and Radius Device.
Date Time Format	Time stamping format for radius transactions.
Location Db Host1	Mongo location for Primary Radius database.
Location Db Host2	Mongo location for Secondary Radius database.
Location Db Port	Port number for the Radius database.
Accounting Enabled	Enables cnAAA to receive incoming Radius Accounting. Default value is True (checked).
Authorization Enabled	Enables cnAAA to receive incoming Radius Authorization. Default value is True (checked).
Coa Enabled	Enables cnAAA to send and receive CoAs.
Disable Location Db	Will not record WLC locations in the Radius mongo DB. Default value is False (unchecked).

RADIUS AAA Proxy Settings

Click RADIUS AAA Proxy Settings to add the configuration in the system. These proxy settings are used for domain-based subscriber authorization.

Table 52: RADIUS AAA Proxy Settings

Parameter	Description
RADIUS Server	Server Identification which will be mapped between Proxy Settings and Domain/Service.
Accounting Port	AAA Server Accounting Port which will receive and process accounting requests.
Authorization Port	AAA Server Authorization Port which will receive and process authentication requests.
Primary IP Address	Primary AAA Server IP address.
Secondary IP Address	Secondary AAA Server IP address.
RADIUS NAS IP Address	NAS IP address which will be sent in the proxied requests.
RADIUS Auth Protocol	RADIUS authentication protocol used. Default: PAP
RADIUS Password	RADIUS authentication password.
Retries	Number of times the requests will be retried in a failure scenario.
Shared Secret	Shared Secret of the AAA Server.
Test User Id	RADIUS username used for testing between cnAAA and AAA Server.
Test Password	RADIUS password used for testing between cnAAA and AAA Server.
Thread Pool Size	Number of threads to handle proxying of requests.
Max Proxy Queue Size	Maximum number of requests that can be queued before being proxied.
Send Test Message	Select this option to send a test message to the AAA server when cnAAA comes up.

Radius Service Template

cnAAA provides reusable and extensible templates for initiating and replying to Radius requests. When the RADIUS plug-in is installed, the Policy Builder includes a section for RADIUS Service Templates within the Reference Data tab. By default, cnAAA includes multiple folders with templates for different access methods.

Create a new RADIUS Service Template

Creating a new RADIUS Service Template called TIMEOUT_ACCESS_ACCEPT based on the existing ISG_ACCESS_ACCEPT template**.**

Procedure

Step 1 In the RADIUS Service Templates panel, click on Summary. Then, click Create Child: RADIUS Service Template Group and name the group "Custom."

Step 2 Select the new, blank Custom group and click **Create Child: RadiusService Template**. Name the new template **TIMEOUT_ACCESS_ACCEPT.**

Step 3 Click the “Select” button next to the Base Template field. Navigate to and select the ISG_ACCESS_ACCEPT template.

Step 4 Expand the “> Show...” dialog under **Show Available AV Pair** Attributes to Add. Type “Cisco” in the Vendors text box, select Cisco, and view available Cisco AVPs. Type **<Radius>** in the Vendors text box, select the <Radius> vendor, and type **IDLE-TIMEOUT** in the Attributes text box. Click the Add button to include the IDLE-TIMEOUT attribute. Repeat to add the SESSION-TIMEOUT attribute.

Step 5 Enter 600 seconds for **IDLE-TIMEOUT** to instruct the ISG on idle session disconnection. Enter 3600 seconds for SESSION-TIMEOUT to set the disconnection time for any session.

The Tag field in the Radius Service Template AV Pair section is deprecated. Do not enter any value.

Step 6 Once the template is created, assign it to a service option using the pick list for the Access Accept template in the Value field.

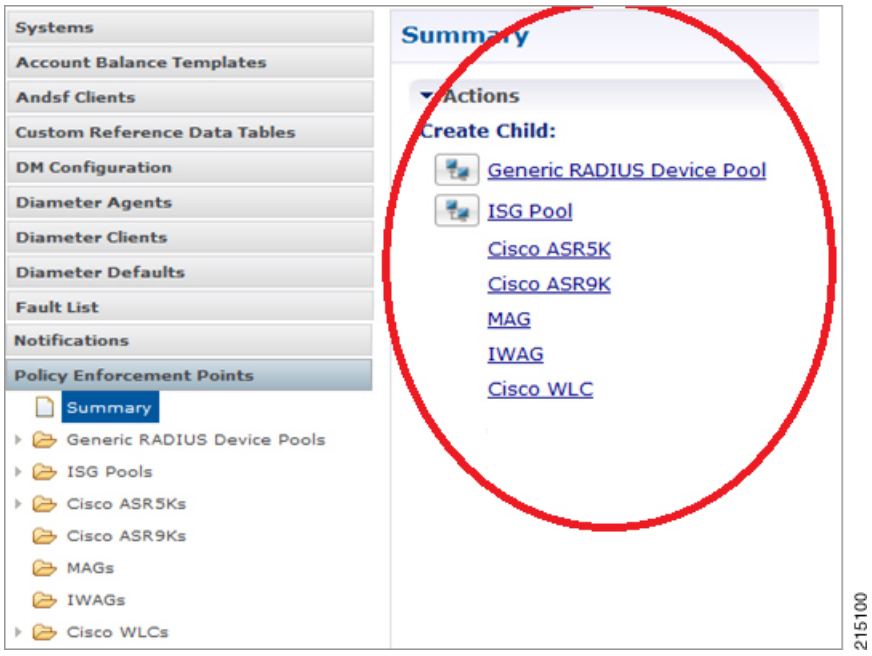
Policy Enforcement Points

A Policy Enforcement Point (PEP) manages policy-based access and acts as a network access system (NAS), though it is not restricted to NAS devices. When a user attempts to access a resource on a network using policy-based access management, the PEP provides the user's attributes to other system components. The PEP delegates decision-making to the Policy Decision Point (PDP), which analyzes applicable policies based on the user's attributes. The PDP decides and returns the result, informing the PEP whether the user is authorized to access the requested resource.

Policy Enforcement Point Tree

After you install Cisco Policy Suite, the Policy Enforcement Points tree in the **Reference Data** tab appears as shown.

Figure 25: Policy Enforcement Point Tree



During installation, you must identify which policy enforcement points your installation uses and which features you need to install.

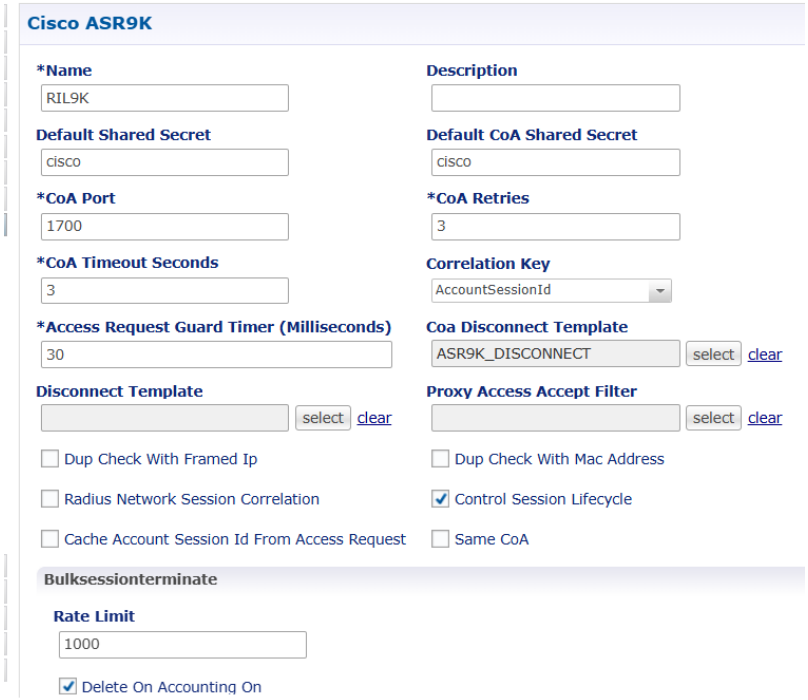
- Generic RADIUS Device Pool
- ISG pool
- Cisco ASR 5K
- Cisco ASR9K
- MAG
- IWAG
- Cisco WLC

Consult the Cisco Technical Representative to configure a custom site.

ASR9K PEP Configuration

ASR9K Policy Enforcement Point (PEP) is used for interfacing CPC with ASR9K devices. PEP configuration for ASR9K is same as Generic Radius device but there is one more additional parameter “Cache Account Session Id from Access Request”. This option stores the value from the Account-Session-Id AVP in the session database during a session.

Figure 26: ASR9K PEP Configuration



To configure ASR9K PEP, follow these steps:

Procedure

Step 1 In the Policy Builder GUI, navigate to **Reference Data** > **System** > **Policy Enforecement Points** > **ASR9K**.

Step 2 To configure the PEP for ASR9K, refer the parameters specified in the ASR9K PEP Configuration image.

Step 3 Click **Save**to submit the parameters.

BNG addition in Policy Builder through external API

The BNG addition feature in the Policy Builder enables you to automatically add devices under the Policy Enforcement Points section. The system provides a REST HTTP interface (GET and POST) to manage BNG IP addresses.

You must publish the Policy Builder before you execute any API requests.

API details and auditing

1. The system logs bng-device-audit.log for the source IP address and the complete request for all addition, modification, and deletion.
2. PB to be published prior hitting the APIs.
3. The headers are uniform for all the functions of the API.

4. Accept header can either be application/xml or text/csv.
5. It is auto-published.

Sample log entry for BNG creation:

```
[rid8749724-smodini-1-worker1/policy-builder/policy-builder-pcf-pcf-engine-app-pcf-green-5dcf45476c-65sb9]
 2026-02-23 08:07:12,158 Source IP: 192.0.2.71, Request Type: CREATE, Repository Name:
anu_import, ASR9K Name: RIL9K, Newly Created BNG Detail: [IP address: 2001:db8:1::77, Shared
 Secret: cisco, CoaSharedSecret: cisco]
```

Configure BNG in PB from external API interface

To perform various actions, follow these steps:

Procedure

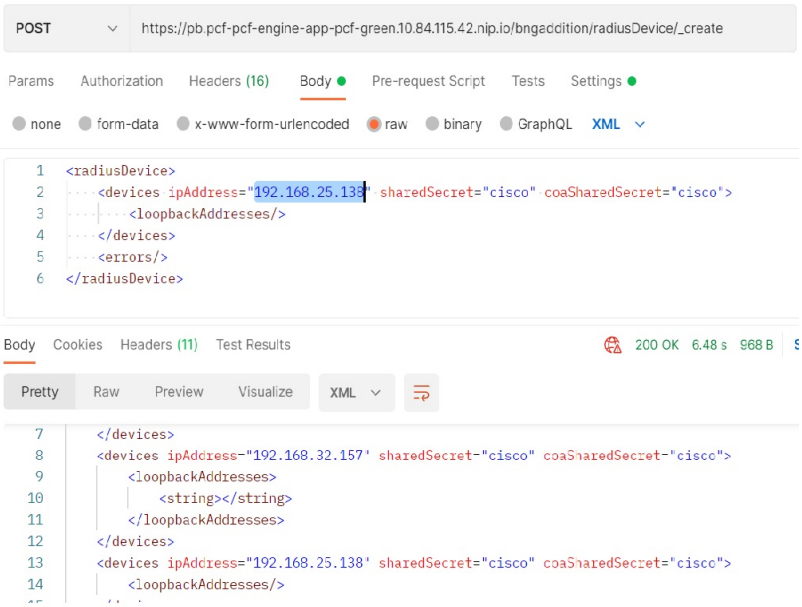
Step 1 Use the HTTP headers for executing automation.

Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
userName	admin
password	Starent@123
repositoryName	TestPB
asr9kName	RIL9K
commitMessage	test
Accept	application/xml

Step 2 Add BNG IP.

URL: https://pb.pcf-pcf-engine-app-pcf-green.10.127.34.151.nip.io/bngaddition/radiusDevice/_create

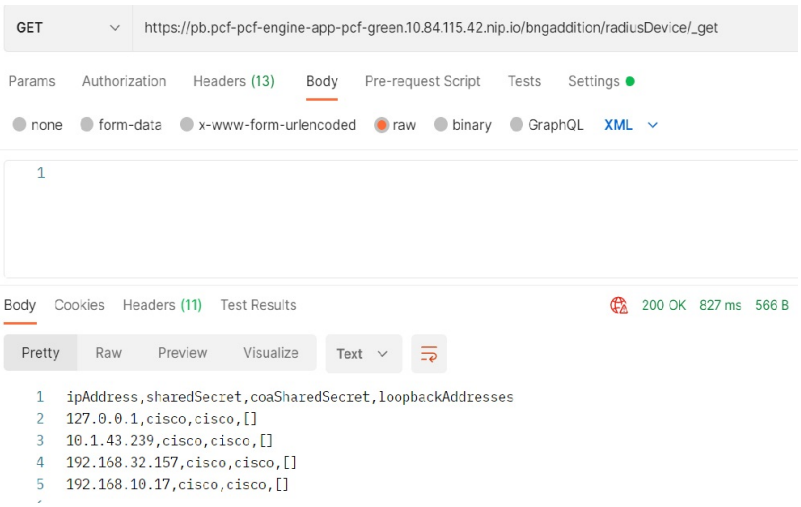
Output:



Step 3 Retrieve BNG IP

URL: https://pb.pcf-pcf-engine-app-pcf-green.10.127.34.151.nip.io/bngaddition/radiusDevice/_get

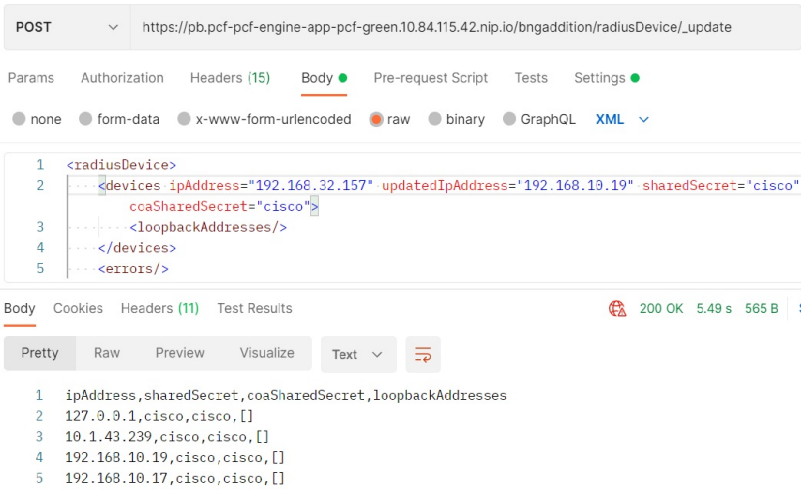
Output:



Step 4 Update BNG IP.

URL: https://pb.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/bngaddition/radiusDevice/_update

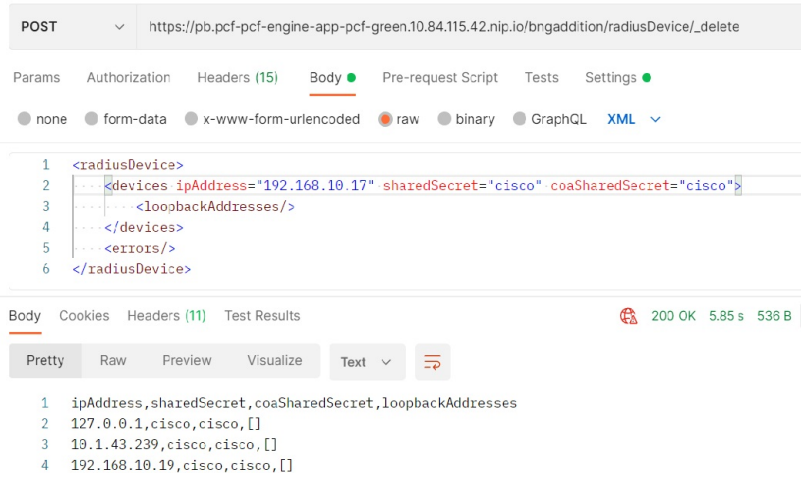
Output:



Step 5 Delete BNG IP.

URL: https://pb.pcf-pcf-engine-app-pcf-green.10.84.115.42.nip.io/bngaddition/radiusDevice/_delete

Output:



Configure BNG in Ops-Center using the RESTCONF API

Use this procedure to manage BNG IP addresses through the RESTCONF API. This allows for automated addition, retrieval, update, and deletion of BNG configurations.

Before you begin

- To obtain the RESTCONF URL from the httproutes output, use this command:

  ```
  kubectl get httproutes -n <namespace> | grep restconf
  ```
- While executing the commands, enter the password when prompted or include it in the command using the `-u "admin:password"` format.

Complete these steps to manage BNG IP addresses:

Procedure

Step 1 Use the `curl -X PATCH` command with the device-group URL to add a new BNG.

Example:

```
curl -X PATCH -H "Accept: application/yang-data+xml" -H "Content-Type: application/yang-data+xml" -k
 https://restconf.ops-center.192.0.2.1.example.com/restconf/data/radius/device-group=ASR9K -u
"admin:password" -d '
<device-group xmlns="http://cisco.com/cisco-mobile-policy-radius">
  <device>
    <name>BNG2</name>
    <ip>192.0.2.138</ip>
    <shared-secret>cisco</shared-secret>
    <coa-shared-secret>cisco</coa-shared-secret>
    <loopback-addresses>192.0.2.2</loopback-addresses>
  </device>
</device-group>'
```

Step 2 Use the `curl -X GET` command to retrieve BNG configurations for the ASR9K device group.

```
curl -X GET -H "Accept: application/yang-data+xml" -H "Content-Length: 0" -k
https://restconf.ops-center.192.0.2.1.example.com/restconf/data/radius/device-group=ASR9K/ -u admin
```

Example:

```
<device-group xmlns="http://cisco.com/cisco-mobile-policy-radius"
xmlns:radius="http://cisco.com/cisco-mobile-policy-radius">
  <name>ASR9K</name>
  <default-shared-secret>$8$a8KhMdDQM7AZzx9jQ51v8aO56QT888AyFQGDNjf5Fqw=</default-shared-secret>
  <default-coa-shared-secret>$8$jjyJ7GsKfao9a0J56JV3VFHXnNsLmgxvTVfaXO9DBPM=</default-coa-shared-secret>

  <coa-port>1700</coa-port>
  <coa-retries>1</coa-retries>
  <coa-timeout-seconds>15</coa-timeout-seconds>
  <device>
    <name>BNG1</name>
    <ip>192.0.2.137</ip>
    <shared-secret>$8$At394wRcyX4qn/BNcqChAzAajOgWwn/Spxdz/Jvsl+E=</shared-secret>
    <coa-shared-secret>$8$RGoJB4uTlQY+4cduTuEBO2qxCS9VNLuQ9ejtWrYXI48=</coa-shared-secret>
    <loopback-addresses>192.0.2.2</loopback-addresses>
  </device>
  <device>
    <name>BNG2</name>
    <ip>192.0.2.138</ip>
    <shared-secret>$8$bYIe6rFZ0CGeyjBuxfyxiprakN6pQ0eK8RLjzSEndIo=</shared-secret>
    <coa-shared-secret>$8$l2bL94SXm3tKQYbyXIuUDmzOVt+kb7QhLN0mRRBUA/8=</coa-shared-secret>
    <loopback-addresses>192.0.2.2</loopback-addresses>
  </device>
</device-group>
```

Step 3 Use the `curl -X PATCH` command to update the IP address for a specific BNG (for example, BNG2).

```
curl -X PATCH \
  -H "Accept: application/yang-data+xml" \
  -H "Content-Type: application/yang-data+xml" \
  -k
https://restconf.ops-center.192.0.2.1.example.com/restconf/data/radius/device-group=ASR9K/device=BNG2/ip
 \
  -u admin \
  -d '<ip xmlns="http://cisco.com/cisco-mobile-policy-radius">192.0.2.139</ip>'
```

Step 4 Use the `curl -X DELETE` command to remove a BNG IP address from the configuration.

```
curl -X DELETE -H "Accept: application/yang-data+json" -k
https://restconf.ops-center.192.0.2.1.example.com/restconf/data/radius/device-group=ASR9K/device=BNG2
 -u admin
```

Sort and search BNG IPs in Policy Builder

Feature history

Feature Name	Release Information	Description
Sort and search BNG IPs in Policy Builder	2026.01.0	This enhancement provides efficient sort and search for BNG IP addresses in Policy Builder (PB). It addresses the challenge of managing large device list. The UI now includes new buttons, a search box, and enhanced REST APIs that provide real-time filtering and sorting for both IPv4 and IPv6 BNG devices.

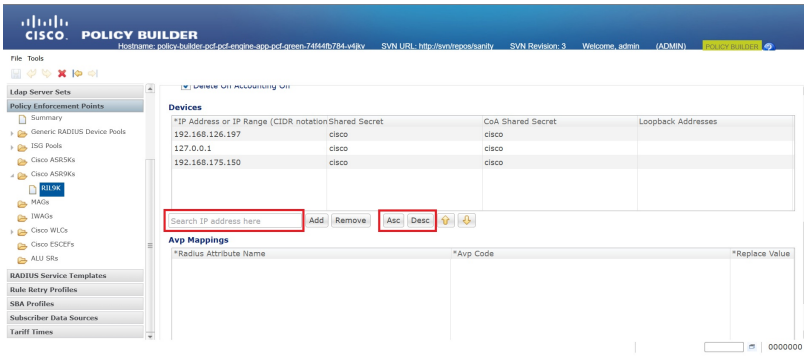
This enhancement introduces sorting and searching for BNG IP addresses within the Policy Builder UI. It simplifies the identification of specific IP addresses in large BNG device lists.

Ascending and **Descending** buttons and a search box enable real-time filtering and ordering of IPv4 and IPv6 addresses in the RADIUS devices table. Additionally, enhanced REST APIs provide sorted device lists to simplify the management of RADIUS policy enforcement points.

Key features and functionality

This feature provides key functionalities to manage BNG IP addresses:

Figure 27: BNG IP address sorting and searching options



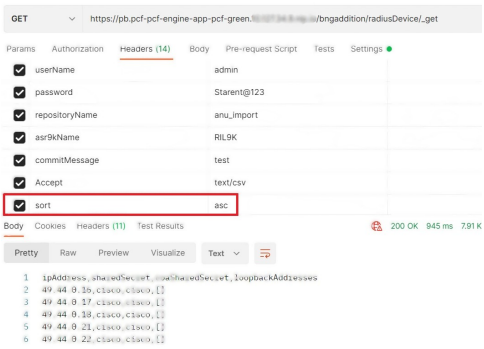
- The **Search Box** updates the device list character-by-character to display matching entries as you type an IP address.

- The **REST APIs** for RADIUS device management now support sorting device lists by ascending or descending order. If no sorting parameter is specified, the output defaults to the original addition time-stamp (the order in which devices were added).

  Supported API Endpoints:

  - Create Device: https://pb.pcf-pcf-engine-app-pcf-green.192.0.2.1.example.com/bngaddition/radiusDevice/_create
  - Get Device List: https://pb.pcf-pcf-engine-app-pcf-green.192.0.2.1.example.com/bngaddition/radiusDevice/_get
  - Update Device: https://pb.pcf-pcf-engine-app-pcf-green.192.0.2.1.example.com/bngaddition/radiusDevice/_update
  - Delete Device: https://pb.pcf-pcf-engine-app-pcf-green.192.0.2.1.example.com/bngaddition/radiusDevice/_delete

Figure 28: API request with sort-order header for device list sorting



Note Use **asc** for ascending order and **desc** for descending order

Known limitations or restrictions

- Only IP addresses can be searched and sorted; other fields, such as shared secrets, are not included.
- If the device list contains more than 1,000 devices, UI-based sorting and searching may experience delays.

Advantages

Validation Prior to Publish in Policy Builder

Validation in the Policy Builder ensures that configuration settings adhere to allowed values before changes are published. During the publishing process, the system automatically identifies unresolved errors, which are configuration issues in the Reference Data, Services, or Policies sections that prevent the Policy Builder from being published. These errors are displayed in a designated section within the Publish window, providing a for network administrators to review.

Although error detection is automated, administrators have the option to manually mask certain acceptable, non-critical errors by creating a "maskPublishErrors.txt" file. This allows them to hide specific errors that do not impact critical operations, facilitating a smoother publishing process.

This validation process ensures that only configurations meeting the required standards are published, maintaining system integrity and functionality.

Multi-User Support

The Policy Builder in the Converged Converged Policy & Charginge supports multi-user functionality, enabling multiple users to work simultaneously on different repositories without interference.

Users can individually update configurations such as Reference Data, Services, or Policies within a repository and save these changes locally. These updates remain private until the user decides to publish them. When another user updates and publishes a separate repository, those changes immediately impact the shared environment, becoming accessible to all users. Thus, changes that are unpublished remain private, while published modifications are shared, facilitating collaborative workflows and ensuring that the shared environment reflects the latest updates. This multi-user support enhances collaboration and efficiency within the network management process.

Support for Export/Import

The Policy Builder in the Converged Converged Policy & Charginge provides support for export and import functionalities, allowing users to efficiently transfer configuration data and maintain consistency across different deployments.

To export a repository from one system, users initiate the export process through the Policy Builder interface, where the system packages the repository data into a ZIP file, ensuring all configurations and settings are included.

This exported repository can then be imported into another Converged Converged Policy & Charginge system. During the import process, users select the exported file and initiate the import through the Policy Builder interface. The system integrates the repository data into the new environment, preserving all configuration details. This functionality facilitates seamless transitions and consistency in configuration management across multiple deployments.

UI changes

Feature History

Feature Name	Release Information	Description
Removing Unused Balance Module Configurations from Control Center and Policy Builder	2026.01.0	This enhancement removes unused Balance module configurations and options from the Control Center (CC) and Policy Builder (PB) graphical user interfaces. It addresses complexity and errors caused by an unutilized module, providing a cleaner, more intuitive interface and preventing mistakes during subscriber management.

This solution simplifies the CPC product by hiding the unused **Balance Module** configurations and its internal modules such as child tabs, system, services menu items from the CC and PB GUIs. This change eliminates several issues:

- Unnecessary Complexity: This refers to the features such as Balance-related tabs, configuration fields, or menu items (configurations tab) that are not relevant to the specific operational needs. In PB, under Account Balance Templates, actions> child tabs are being hidden form the UI.
- Potential User Errors: Hiding the **Balance Module** prevents incorrect data entry into irrelevant fields, or general confusion from interacting with features that have no functional purpose in the specific deployment.

By removing these non-essential elements, the goal is to provide a cleaner, more intuitive interface. The system achieves this by hiding both **UI components** (such as child tabs, system, services menu items of the PB and CC) from the CPC product.

- Policy Builder: Modifying the **internal models (such as child tabs, system, services menu items)** to eliminate Balance-related options. In PB, under Account Balance Templates, actions> child tabs are being hidden form the UI.
- Control Center: Removing the **specific user interface components such as Balance tab under subscriber profile** that supported the Balance module.

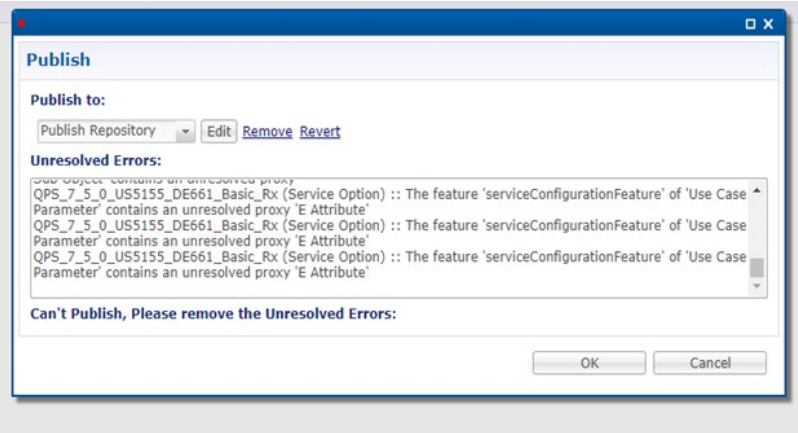
Unresolved errors display during publish

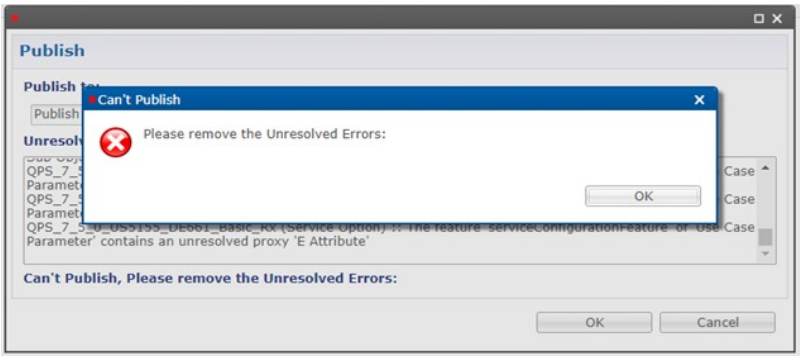
This feature provides a summary of configuration errors in the **Publish** window of the **Policy Builder**. When you attempt to publish with configuration errors, an error window appears to notify you of the issues. If there are no unresolved errors, the **Publish** window opens normally.

Enable or disable unresolved error display

Use this Ops-Center configuration command to enable unresolved errors window.

- In the `engine <engine group name> policy-builder properties com.broadhop.unresolvedError.featurevalue` *`true`* `exit`

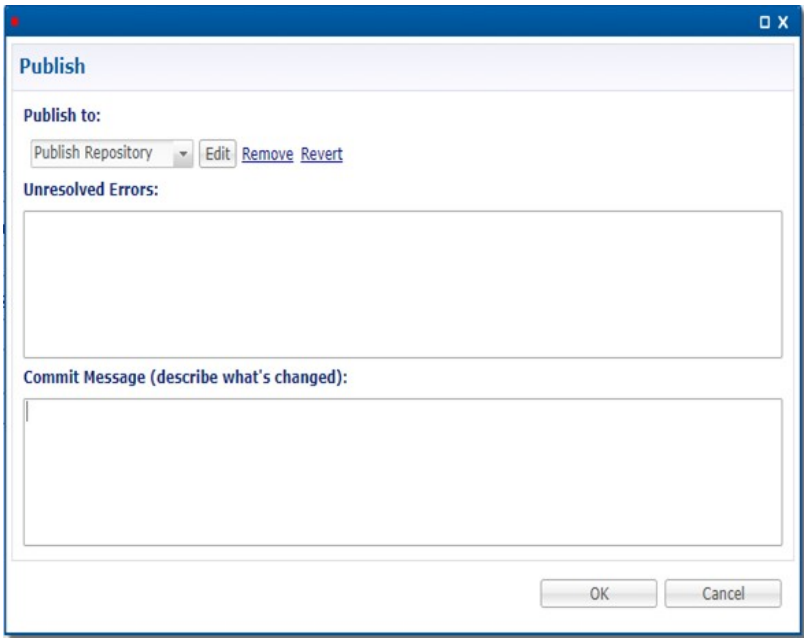




Disable unresolved error display

Use this Ops-Center configuration command to disable the unresolved error window during the publishing process.

- In the `engine <engine group name> policy-builder properties com.broadhop.unresolvedError.featurevalue` *`false`* `exit`



Manage OCS and BNG configurations with single API

Feature description

The CPS includes a RESTful API (deletion) and automation scripts that manage OCS server and BNG device details. This feature replaces manual configuration processes in Policy Builder and Ops-Center. As a result, deployment time is reduced and the risk of human error decreases.

Prerequisites

Run these scripts during the maintenance window. Executing this action during active traffic can cause traffic fluctuation.

Configure the Ops-Center properties for API access

Procedure

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

Step 2 Configure the Policy Builder (PB) properties.

```
engine <engine_name>
  policy-builder opscenter-password <password>
  policy-builder properties opsCenterUrl value <opscenter_url>
  policy-builder properties opsCenterUsername value <username>
  policy-builder properties enableDeleteOperationForBngDevices value false
  policy-builder properties enableDeleteOperationforRadiusServer value false
exit
```

Note The PB pod restarts after you update the properties.

Example:

```
engine pcf-green(name of the Engine)
 policy-builder properties enableDeleteOperationForBngDevices
  value false
 exit
policy-builder properties enableDeleteOperationforRadiusServer
  value false
 exit
 policy-builder properties opsCenterUrl
  value https://restconf httproutes url
 exit
 policy-builder properties opsCenterUsername
  value <username>
 exit
policy-builder opscenter-password <password>
 exit
```

Step 3 `commit` the changes.

Note

a. If Ops-Center is not configured, provide the details in the interactive prompt. Otherwise, the devices are added only to PB and not to Ops-Center.

b. The `enableDeleteOperationForBngDevices` and `enableDeleteOperationforRadiusServer` properties control the deletion endpoints. If these properties are set to `false` (the default value), the system blocks all delete requests and returns an HTTP 4xx Forbidden error.

You can copy the script from this location: `/data/<name space>/data-pcf-utilities-0/support/script/bng_device_ocs_server`. The script folder contains sample input files for BNG devices and radius server format for reference.

BNG device additions to the PB and Ops-Center

The `add_bng_devices_bulk.py` script utilizes this command-line syntax to automate the provisioning of BNG devices:

```
python add_bng_devices_bulk.py --input <file> [OPTIONS]
```

Table 53: Command Line Arguments

Argument	Default	Description
`--input, -i`	`(required)`	Path to the CSV/TXT/JSON file of device IPs
`--group-name`	`ASR9K`	Device group name in Policy Builder and Ops-Center

Argument	Default	Description
`--batch-size`	`1000`	Number of devices per API call
`--timeout`	`600`	Per-batch HTTP timeout in seconds
`--yes`	`false`	Skip the YES confirmation prompt
`--skip-invalid`	`false`	Skip invalid rows instead of aborting
`--rejects-file`	`<inputfilename>.rejected`	Path for the rejected-rows output file
`--details-file`	(stdout)	Write full duplicate/failure lists to a file

Script asks for these details as interactive prompt:

Table 54: Interactive prompts

Prompt	Example value	Notes
API host	`pb.pcf-pcf-engine-app-pcf-green.<host-ip>.nip.io`	https:// prepended automatically
PB API username	`<pb-username>`	Required
PB API password	(hidden)	Required
opsCenterUsername	`<ops-center-username>`	Optional — blank = use PB engine property
opsCenterPassword	(hidden)	Optional — blank = use PB engine property
sharedSecret	(hidden, confirmed twice)	Required
coaSharedSecret	(hidden, confirmed twice)	Required
repositoryUrl	`http://svn/repos/<your-repo>`	Required
opsCenterUrl	`https://restconf.pcf-ops-center.<host-ip>.nip.io`	Optional — blank = use PB engine property

Note When configured, the script retrieves the `opsCenterUrl`, `opsCenterUsername`, and `opsCenterPassword` from the Ops-Center Policy Builder (PB) engine properties. The script also adds `https://` to the `opsCenterUrl` if the no scheme is specified.

Execute the BNG Bulk addition script

The `add_bng_devices_bulk.py` script automates the provisioning of BNG devices. To execute the script, use this command structure:

```
python add_bng_devices_bulk.py --input <filename> --skip-invalid --details-file
<output_filename>
```

Example:

```
python add_bng_devices_bulk.py --input bng.csv --skip-invalid --details-file
      bng_details.txt
```

Command parameters:

- `--input`: Specifies the path to your source file (CSV, TXT, or JSON).
- `--skip-invalid`: Instructs the script to bypass invalid rows rather than aborting the entire process.
- `--details-file`: Defines the file where the script writes comprehensive logs regarding duplicates or failures.

Table 55: Execution summary

Field	Description
Skipped (invalid)	Number of rows excluded due to formatting or validation errors.
Input duplicates	Number of identical IPs found within the source file; only the first is processed.
Sent to API	Total number of valid device entries transmitted to the API.
System duplicates	Devices that already existed in the Policy Builder (PB) or Ops-Center.
Succeeded	Total count of devices successfully added or synced.
Failed	Number of devices that could not be processed due to errors.
Elapsed	Total time taken to complete the operation (in seconds).

The final result status (e.g., `PARTIAL` or `SUCCESS`) indicates the overall health of the batch. If the status is `PARTIAL`, review the file specified in the `--details-file` argument to identify and resolve specific device failures.

Note

- The RADIUS pod restarts following a successful batch, it is recommended to perform these BNG addition or deletion during a scheduled maintenance window.
- For detailed tracking, verify `bng-device-audit.log` located in the `/data/consolidated-aaa-logging/` directory on the `consolidated-aaa-logging-0` pod.

Add Bulk BNG Device

1. Copy the script `add_bng-devices_bulk.py` from `/data/<name space>/data-pcf-utilities-0/support/script/bng_device_ocs_server` to the master node.

2. Execute this script to avoids invalid IP addresses in the input file.

```
python add_bng-devices_bulk.py --input bng.csv --skip-invalid
```

3. Enter inputs in the interactive prompt.

```
BNG Addition Interactive Prompt:
API host:<PB httproutes URL>
PB username (for HTTP Basic auth):<username>
PB password:
opsCenterUsername [blank = use PB engine property]:
opsCenterPassword [blank = use PB engine property]:
sharedSecret (case-sensitive, blank = empty):
coaSharedSecret (case-sensitive, blank = empty):
repositoryUrl:<PB repo URL>
opsCenterUrl [blank = use PB engine property]:

Target host : <PB httproutes URL>
Total devices : 50
Type "YES" to proceed, anything else to abort: yes
```

4. After confirmation, script execution starts and you will see one of these scenarios.
   - Scenario 1: All devices succeed

```
1/1] POSTing 50 devices to https://<api-host>/api/bng/device/bulk ...

-> total devices=50, succeeded=50, failed=0, alreadyExists=0

=================== Bulk Add Summary ===================

Input duplicates : 0 (same IP repeated in file; first occurrence kept, the rest
skipped locally)

Sent to API : 50

System duplicates : 0 (already existed in PB / Ops-Center)

Succeeded : 50

Failed : 0

Elapsed (seconds) : 12.1

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

Result: SUCCESS - 50 device(s) succeeded.
```

   - Scenario 2: The process was run partially with the `--skip-invalid` flag. Some items were invalid, some already existed, and one failed.

```
[1/1] POSTing 47 devices to https://<api-host>/api/bng/device/bulk ...

-> total devices=47, succeeded=42, failed=1, alreadyExists=4

=================== Bulk Add Summary ===================

Skipped (invalid) : 2 (see devices.txt.rejected)

Input duplicates : 1 (same IP repeated in file; first occurrence kept, the rest
skipped locally)

Sent to API : 47

System duplicates : 4 (already existed in PB / Ops-Center)
```

```
Succeeded : 42

Failed : 1

Elapsed (seconds) : 18.7

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

Result: PARTIAL - 42 succeeded, 4 already existed, 1 failed (see below).

======================================================

System duplicates - already existed (4 total, showing 4):

- BNG-198.18.0.1 (198.18.0.1)

- BNG-198.18.0.2 (198.18.0.2)

- BNG-198.18.0.3 (198.18.0.3)

- BNG-198.18.0.4 (198.18.0.4)

Failures (1 total, showing 1):

- BNG-198.18.0.99 (198.18.0.99): IP Address range conflicts with other IP range or
IP provided

```

```
> **`Skipped (invalid)`** only appears when `--skip-invalid` is used. In a plain
run, any invalid row aborts the script before any API call is made.
```



Note To verify the BNG device summary, check `bng-device-audit.log` file located in `/data/consolidated-aaa-logging/` on the `consolidated-aaa-logging-0` pod.

While adding multiple BNG devices, it is recommended to add batch size (example: --batch-size 5) to avoid 504 gateway issue.

Verify the changes in PB and Ops-Center

Use this procedure to confirm that your configuration is applied successfully.

1. Log in to the Policy Builder UI.
2. Navigate to **Policy Enforcement Points** > **Cisco ASR9Ks** >**RIL9K** and verify the BNG IPs under **Devices** section.
3. Log in to the Ops-Center to verify that the BNG device details:

   ```
   pcf# show running-config radius device-group ASR9K device
   ```

Table 56: Invalid IP error handling

Mode	Behaviour

Default (fail-fast)	If any IP or loopback is invalid, the script prints all bad rows and exits before making any API call. Fix the file and re-run.
--skip-invalid	Invalid rows are skipped and written to --rejects-file. Valid rows are sent. The rejects file is a valid input file for re-run.
--rejects-file <path>	Implies --skip-invalid. Overrides the default rejects filename.

Delete BNG devices using the REST API

Use this procedure to delete individual or multiple BNG devices from the configuration using the REST API.

- Delete a single BNG device: Submit a `POST` request to the `/api/bng/device/delete` endpoint.

```
curl -k -u admin:<password> -H "Content-Type: application/json" -X POST
"https://<pb_httproutes_url>/api/bng/device/delete" -d
'{"ip":"192.0.2.0/18","groupName":"ASR9K","repositoryUrl":"http://svn/repos/Test_PB"}'
```

  Example:

```
{"status":"success","ipAddress":"192.0.2.0/18","groupName":"ASR9K",
"policyBuilder":{"status":"success"},"opsCenter":
{"status":"success","deviceName":"BNG-192.0.2.0/18"}}
```

- Delete BNG devices in bulk: Submit a `POST` request to the `/api/bng/device/bulk/delete` endpoint.

```
curl -k -u admin:<password> -H "Content-Type:
application/json" -X POST "https://<pb_httproutes_url>/api/bng/device/bulk/delete"
-d '{"ips":["192.0.2.1","192.0.2.0/16","2001:db8::1"],"groupName":"ASR9K",
"repositoryUrl":"http://svn/repos/Test_PB"}'
```

  Example:

```
{
  "status": "success",
  "summary": {
    "total": 3,
    "success": 3,
    "notFound": 0,
    "partialSuccess": 0,
    "failed": 0
  },
  "results": [
    {"ipAddress":"192.0.2.1","deviceName":"BNG-192.0.2.1","status":"success",
"policyBuilder":{"status":"success"},"opsCenter":{"status":"success"}},
    {"ipAddress":"192.0.2.0/16","deviceName":"BNG-192.0.2.0/16","status":"success",
"policyBuilder":{"status":"success"},"opsCenter":{"status":"success"}},
    {"ipAddress":"2001:db8::1","deviceName":"BNG-2001:db8::1","status":"success",
"policyBuilder":{"status":"success"},"opsCenter":{"status":"success"}}
  ]
}
```

Radius server addition in PB and Ops-Center

The `add_radius_servers_bulk.py` script automates the bulk provisioning of RADIUS AAA Proxy settings into the Policy Builder (PB) and Ops-Center through the REST API. When the script is executed, it shows a categorized summary that describes the status of each server, such as Added to PB, Added to Ops-Center, or Failed.

Mandatory input columns for RADIUS server provisioning

To provision RADIUS servers using the `add_radius_servers_bulk.py` script, the input file (CSV or JSON) must contain these mandatory columns. Each of these fields must be non-empty for every row in the input file.

`data/<pcf-namespace>/data-pcf-utilities-0/support/script/bng_device_ocs_server/`

Table 57: Mandatory input columns

Column	UI Label	Example Value
`radiusServer`	Radius Server	DEEL1_OCS
`authorizationPort`	Authorization Port	1812
`accountingPort`	Accounting Port	1813
`primaryIpAddress`	Primary IP Address	10.1.43.75
`radiusAuthProtocol`	RADIUS Auth Protocol	PAP
`sharedSecret`	Shared Secret	REPLACE_ME_SHARED_SECRET
`retries`	Retries	3
`timeoutSeconds`	Timeout Seconds	5
`threadPoolSize`	Thread Pool Size	10
`maxProxyQueueSize`	Max Proxy Queue Size	100

Use this syntax to execute the RADIUS server addition script: `python add_radius_servers_bulk.py --input <file> [OPTIONS]`

Table 58: Command line arguments

Argument	Default	Description
`--input, -i`	Required	Path to the CSV or JSON input file.
`--batch-size`	50	Number of servers processed per API call.
`--timeout`	600	HTTP timeout duration per batch (in seconds).
`--yes`	false	Skips the manual confirmation prompt.
`--details-file`	stdout	Path to save full duplicate/failure lists.
`--default-<field>`	Varies	Sets default values for optional CSV columns (e.g., `--default-retries 3`).

Interactive configuration

When executing the script, you will be prompted to provide these details. If Ops-Center , username, password, and URL are already configured in the Policy Builder (PB) engine properties, you may leave these values blank.

Prompt	Example Value	Notes
API host	pb.pcf-pcf-engine-app-pcf-green.<host-ip>.nip.io	https:// is prepended automatically.
PB API username	<pb-username>	Required; uses HTTP Basic authentication.
PB API password	(hidden)	Required.
repositoryUrl	http://svn/repos/<your-repo>	Required; defaults to file value if left blank.
opsCenterGroup	grp1 or grp1,grp2	Blank = no sync; defaults to file value if left blank.
opsCenterUrl	https://restconf.pcf-ops-center.<host-ip>.nip.io	Optional; defaults to PB property or file value.
opsCenterUsername	<ops-center-username>	Optional; defaults to PB property or file value.
opsCenterPassword	(hidden)	Optional; defaults to PB property or file value.

You can customize the `add_radius_servers_bulk.py` script using various command-line flags to manage provisioning workflows.

Table 59: Execution Summary

Option	Command Example	Description
Basic Run	`python add_radius_servers_bulk.py --input radius_servers.csv`	Executes the script with default settings.
JSON Input	`python add_radius_servers_bulk.py --input radius_servers.json`	Processes RADIUS servers from a JSON file.
Batch Sizing	`python add_radius_servers_bulk.py --input radius_servers.csv --batch-size 10`	Reduces the number of servers per API call.
Custom Defaults	`python add_radius_servers_bulk.py --input servers.csv --default-auth-protocol PAP --default-retries 5 --default-timeout-seconds 10`	Applies default values for optional CSV columns.
Capture Details	`python add_radius_servers_bulk.py --input radius_servers.csv --details-file results.csv`	Saves a full list of duplicates or failures to a file.
Non-Interactive	`python add_radius_servers_bulk.py --input servers.csv --yes`	Skips confirmation prompts for automated execution.
Timeout Adjust	`python add_radius_servers_bulk.py --input devices.txt --timeout <seconds>`	Increases the timeout duration for large-scale additions.

Add Bulk RADIUS Server

1. Copy the script `add_radius_servers_bulk.py` from `/data/<name space>/data-pcf-utilities-0/support/script/bng_device_ocs_server` to the master node.
2. Execute this script:

   ```
   python add_radius_servers_bulk.py --input radius_servers.csv
   ```

3. Enter inputs in the interactive prompt.

   ```
   === Bulk RADIUS Server Add ===
   API host (e.g. pb.pcf-pcf-engine-app-pcf-green.10.84.115.53.nip.io): <PB httproutes URL>
   PB username (for HTTP Basic auth): <username>
   PB password:
   repositoryUrl (e.g. http://svn/repos/test) (blank=use values from file):
   opsCenterGroup (e.g. grp1 or grp1,grp2) [blank = no Ops-Center sync] (blank=use values
   from file):
   opsCenterUrl [blank = use PB engine property]:
   opsCenterUsername [blank = use PB engine property]:
   opsCenterPassword [blank = use PB engine property]:
   Target host : <PB httproutes URL>
   Total servers : 7
   Batches     : 1
   Type "YES" to proceed, anything else to abort:yes
   ```

4. After confirmation, script execution starts and you will see one of these scenarios.
   - Scenario 1: All devices succeed

     ```
     ================== Bulk Add Summary ==================
       Input duplicates    : 1    (radiusServer names repeated in file)
       Sent to API         : 7
       Added to PB         : 5
       Already in PB       : 1    (unique names already present before this run)
          - in PB + Ops-Center : 1
          - in PB only         : 0
       Duplicate rows      : 1    (same name repeated in input; counted once above)
       Added to Ops-Center : 5
       Failed in PB        : 0
       Failed in Ops-Center: 0    (added to PB but NOT to Ops-Center)
       Elapsed (seconds)   : 43.3
     ------------------------------------------------------
       Result: SUCCESS - 5 new server(s) added; 1 already existed. (1 duplicate row(s)
     in input skipped.)
     ======================================================
     Details written to: ocs_details.txt
     ```

Note While adding multiple RADIUS servers, it is recommended to add batch size (example: --batch-size 5) to avoid 504 gateway issue.

The RADIUS pod restarts following a successful batch, it is recommended to perform these RADIUS server addition or deletion during a scheduled maintenance window.

For detailed audit logs regarding these additions, refer to the `radius-aaa-audit.log` file located in `/data/consolidated-aaa-logging/` on the `consolidated-aaa-logging-0` pod.

Verify the changes in PB and Ops-Center

Use this procedure to confirm that your configuration is applied successfully.

1. Log in to the Policy Builder UI.
2. Navigate to **Systems** > **system-1** >**Plugin Configurations**> **RADIUS Configuration** and verify RADIUS server details.
3. Log in to the Ops-Center to verify that the RADIUS server details:

   ```
   pcf# show running-config radius server-group grp1 servers
   ```

Error handling

The `add_radius_servers_bulk.py` script performs comprehensive client-side validation on all input rows before initiating any HTTP requests. If the script detects an error, it will abort the process to prevent partial or incorrect configurations.

If a validation error occurs, the script will not send any data to the API. You must correct the reported issues in your input file and re-run the script.

Error Condition	Error Message Format
Missing mandatory field	`ERROR parsing <file>: line N: missing mandatory field(s): <name>`
Invalid integer field	`line N: Authorization Port (authorizationPort)='abc' is not an integer`
Invalid IP field	`line N: Primary IP Address (primaryIpAddress)='hostname' is not a valid IP`
Invalid JSON format	`ERROR: JSON input must be an array of server objects`

Delete RADIUS server REST API

- Single Deletion: Send a `DELETE` request to:
  `https://<PB_httproutes_URL>/api/repository/actions/radius/<radiusServer>?repositoryUrl=<PB repo URL>&opsCenterGroup=grp1`

  Example:

```
{
  "status": "success",
  "message": "RADIUS AAA Proxy Settings deleted successfully",
  "id": "DummyOCS1",
  "pbStatus": "deleted",
  "cascadeDelete": {
    "radiusServer": "deleted"
  },
  "opsCenter": {
    "grp1": {
      "status": "deleted"
    },
    "summary": {
      "success": 1,
      "failed": 0
    }
  }
}
```

- Bulk Deletion: Send a `DELETE` request with a JSON array of objects to:

```
https://<PB_httproutes_URL>/api/repository/actions/radius/bulk?repositoryUrl=<PB repo
URL>" -H "Content-Type: application/json" -d '
[{"radiusServer":"DummyOCS2","opsCenterGroup":"grp1"},{"radiusServer":"DummyOCS3","opsCenterGroup":"grp1"},{"radiusServer":"DummyOCS4",
"opsCenterGroup":"grp1"},{"radiusServer":"DummyOCS5","opsCenterGroup":"grp1"}]'
```

Example:

```
{
  "summary": {
    "total": 4,
    "deleted": 4,
    "notFound": 0,
    "pbFailed": 0,
    "opsCenterDeleted": 4,
    "opsCenterFailed": 0,
    "opsCenterSkipped": 0
  },
  "results": [
    {
      "radiusServer": "DummyOCS2",
      "pbStatus": "deleted",
      "status": "deleted",
      "opsCenter": {
        "grp1": {
          "status": "deleted"
        },
        "summary": {
          "success": 1,
          "failed": 0
        }
      }
    },
    {
      "radiusServer": "DummyOCS3",
      "pbStatus": "deleted",
      "status": "deleted",
      "opsCenter": {
        "grp1": {
          "status": "deleted"
        },
        "summary": {
          "success": 1,
          "failed": 0
        }
      }
    },
    {
      "radiusServer": "DummyOCS4",
      "pbStatus": "deleted",
      "status": "deleted",
      "opsCenter": {
        "grp1": {
          "status": "deleted"
        },
        "summary": {
          "success": 1,
          "failed": 0
        }
      }
    },
    {
      "radiusServer": "DummyOCS5",
      "pbStatus": "deleted",
```

```
        "status": "deleted",
        "opsCenter": {
          "grp1": {
            "status": "deleted"
          },
          "summary": {
            "success": 1,
            "failed": 0
          }
        }
      }
    ]
}
```



Note Ensure that the `repositoryUrl` and `opsCenterGroup` parameters are correctly specified to ensure synchronization between the Policy Builder and Ops-Center.

SOAP API engine rate limiting

SOAP API engine rate limiting protects the cnAAA Engine from excessive SOAP API POST traffic. The mechanism rejects any requests that surpass the configured TPS threshold.

The feature uses these Policy Builder properties:

- The `maxApiTpsThreshold` parameter sets the maximum number of SOAP API transactions permitted per second. If you set this value to `0`, requests are allowed without throttling.
- When the threshold is exceeded, `httpErrorCodeOnThresholdReached` returns the rejection code.

After Policy Builder publishes a configuration change, the cnAAA Engine updates the rate-limiting parameters without an engine restart.

You can monitor rate-limiting performance using the following methods:

- Use the `api-ratelimiting-reject-429` KPI to track specific requests rejected after reaching the configured rate limit.
- Use Grafana integrations for system monitoring and visibility.



Note This feature applies rate limiting only to SOAP API POST requests. It does not limit REST GET requests.

Configure SOAP API engine rate limiting

Use this procedure to configure the TPS threshold and the rejection code for SOAP API requests in the Policy Builder. This enables dynamic rate limiting and threshold-based rejection without requiring a cnAAA engine restart.

Before you begin

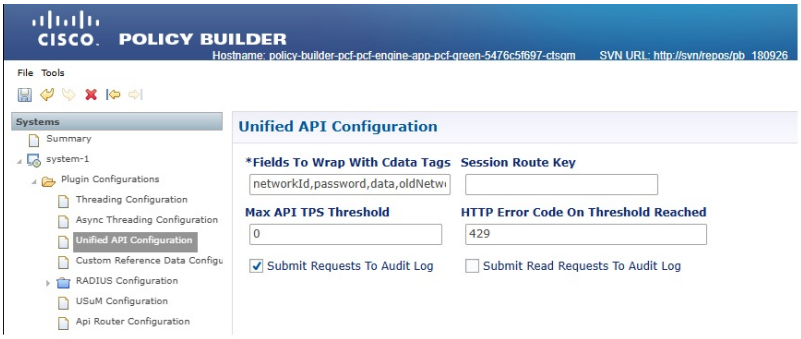
- Ensure the CPC deployment includes the SOAP API engine rate-limiting feature.
- Verify that the Policy Builder is integrated with the cnAAA engine.
- Ensure you have administrative access to the Policy Builder.
- Identify the rejection code required for your deployment.
- Ensure that Grafana integrations are configured if you require KPI monitoring.

Procedure

Step 1 Log in to the Policy Builder

Step 2 Navigate to the Unified API configuration page. Open the configuration that contains the SOAP API rate-limiting properties.

Figure 29: Unified API configuration in Policy Builder



Step 3 Set the maximum API TPS threshold. In the Max API TPS Threshold field, enter the desired threshold.

Example:

```
maxApiTpsThreshold=1000
```

Set the value to `0` to allow requests without throttling, or set a value greater than `0` to enable threshold-based rejection.

Step 4 Set the HTTP error code for threshold violations. In the HTTP Error Code On Threshold Reached field, enter the rejection code supported by your deployment.

Example:

```
httpErrorCodeOnThresholdReached=429
```

Use `429`. Replace it with the rejection code supported by your deployment.

Step 5 Publish the Policy Builder configuration. Choose **File** > **Publish to Runtime**.

Step 6 Send SOAP API POST requests below and above the configured threshold.

- Requests below the threshold must be processed.
- Requests above the threshold must be rejected with the configured rejection code.

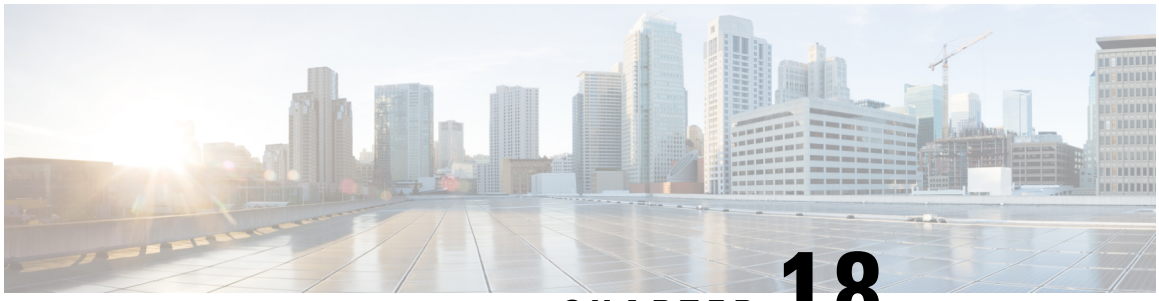
Step 7 Verify the SOAP API rejection KPI.

- Confirm that the `api-ratelimiting-reject-429` KPI accurately records requests rejected after the threshold is exceeded.
- Verify the rejection event through Grafana integrations.

Step 8 Confirm that SOAP API processing continues after the configuration update.

Confirm that no cnAAA Engine restart is required.

The cnAAA Engine applies the published SOAP API rate-limiting values. It rejects requests above the configured threshold. It also records threshold-based rejections for monitoring, without requiring an engine restart.



CHAPTER 18

Plugin

- Overview, on page 365
- Threading Configuration, on page 366
- RADIUS Configuration, on page 368
- RADIUS AAA Proxy Settings, on page 370
- ASR9K Configuration, on page 371

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.

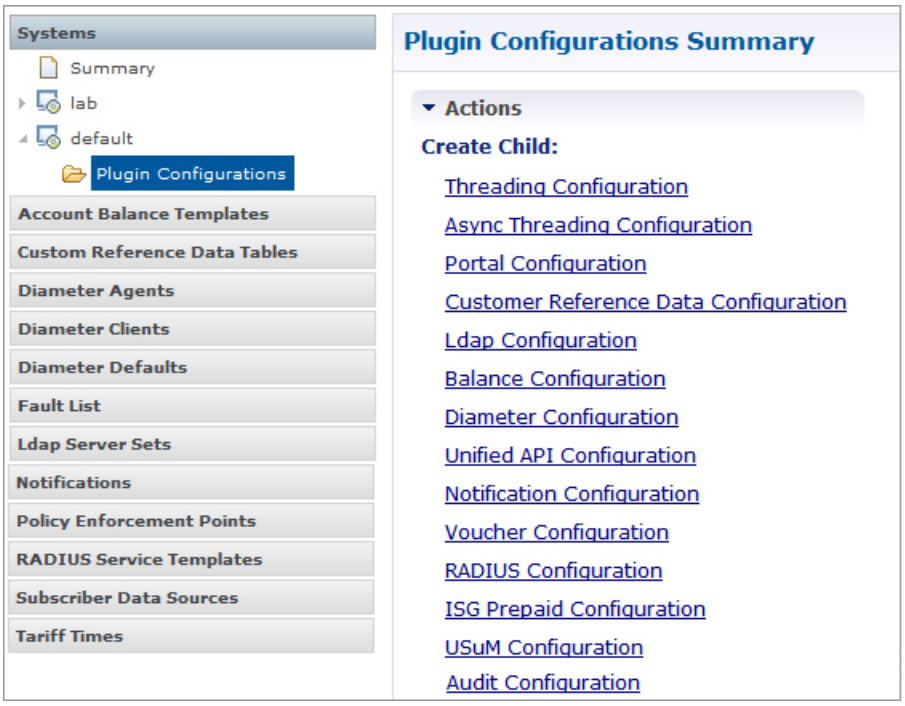
CPC provides core plug-ins for customizing and optimizing your installation.

- 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 30: 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.

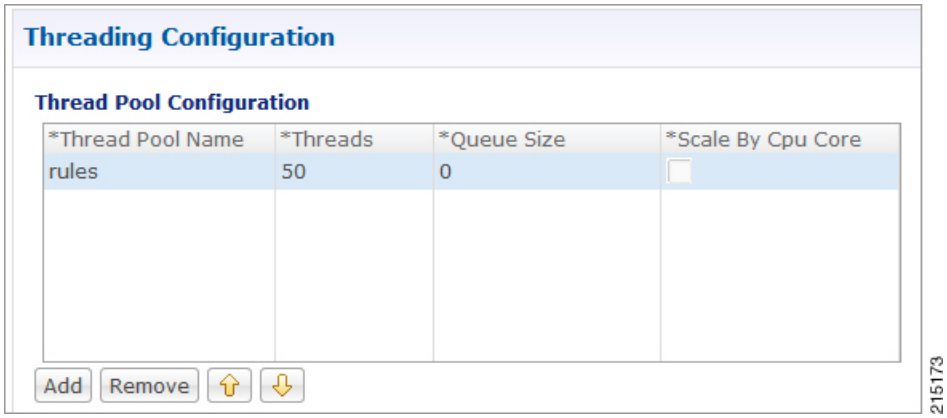
Click **Threading Configuration** in the right pane to add the threading configuration to the system. This is a performance tuning parameter and can be changed in case of a performance issue according to the call model and hardware. For further information, contact your Cisco Account representative.

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 31: Thread Pool Configuration



The following parameters can be configured under Threading Configuration:

Table 60: Threading Configuration Parameters

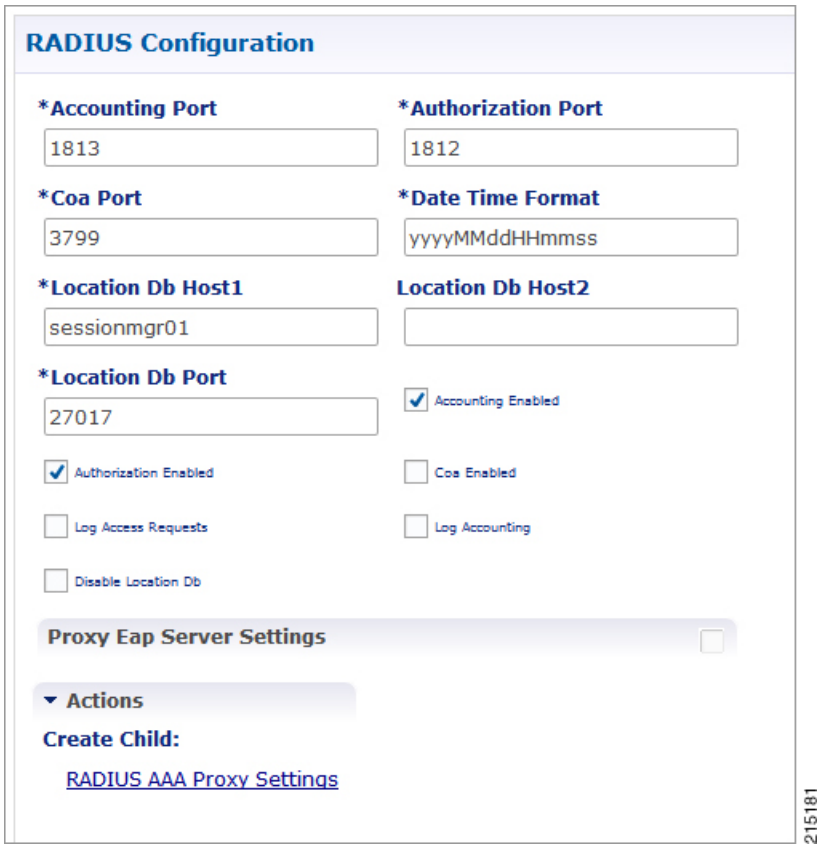
Parameter	Description
Thread Pool Name	Name of the Cisco thread pool i.e., rules.
Threads	Specify the threads to set in the thread pool. You can set rules thread ranging from 50 to 100 depending on the call flow (based on number of lookup operations). • rules = 50; Queue Size = 0; Scale By Cpu Core = 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.

Parameter	Description
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.
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.

RADIUS Configuration

Click **RADIUS Configuration** in the right pane to add the configuration in the system.

Figure 32: RADIUS Configuration



The following parameters can be configured under RADIUS Configuration:

Table 61: RADIUS Configuration Parameters

Parameter	Description
Accounting Port	Port used for incoming radius accounting.
Authorization Port	Port used for incoming radius authorization.
Coa Port	Port used for Change of Authority between CPC and Radius Device.
Date Time Format	Time stamping format for radius transactions.
Location Db Host1	Mongo location for Primary Radius database.
Location Db Host2	Mongo location for Secondary Radius database.
Location Db Port	Port number for the Radius database.
Accounting Enabled	Enables CPC to receive incoming Radius Accounting. Default value is True (checked).

Parameter	Description
Authorization Enabled	Enables CPC to receive incoming Radius Authorization. Default value is True (checked).
Coa Enabled	Enables CPC to send and receive CoAs.
Log Access Requests	Log the radius accounting which is configured in `/etc/broadhop/logback.xml`. The typical default logging location is `/var/broadhop/radius/accounting/accounting.current`.
Log Accounting	Logs radius authorization requests, also configured in `/etc/broadhop/logback.xml`. The typical default logging location is `/var/broadhop/radius/access/rejects.current`.
Disable Location Db	Will not record WLC locations in the Radius mongo DB. Default value is False (unchecked).

For information on proxy settings, refer to RADIUS AAA Proxy Settings.

RADIUS AAA Proxy Settings

Click **RADIUS AAA Proxy Settings** to add the configuration in the system. These proxy settings are used for domain-based subscriber authorization.

Table 62: RADIUS AAA Proxy Settings

Parameter	Description
RADIUS Server	Server Identification which will be mapped between Proxy Settings and Domain/Service.
Accounting Port	AAA Server Accounting Port which will receive and process accounting requests.
Authorization Port	AAA Server Authorization Port which will receive and process authentication requests.
Primary IP Address	Primary AAA Server IP address.
Secondary IP Address	Secondary AAA Server IP address.
RADIUS NAS IP Address	NAS IP address which will be sent in the proxied requests.
RADIUS Auth Protocol	RADIUS authentication protocol used. Default: PAP
RADIUS Password	RADIUS authentication password.
Retries	Number of times the requests will be retried in a failure scenario.
Shared Secret	Shared Secret of the AAA Server.

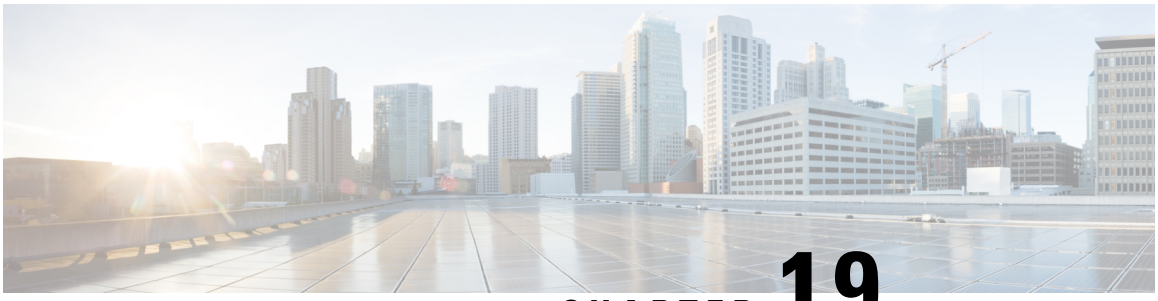
Parameter	Description
Test User Id	RADIUS username used for testing between CPC and AAA Server.
Test Password	RADIUS password used for testing between CPC and AAA Server.
Thread Pool Size	Number of threads to handle proxying of requests.
Max Proxy Queue Size	Maximum number of requests that can be queued before being proxied.
Send Test Message	Select this option to send a test message to the AAA server when CPC comes up.

ASR9K Configuration

Click **ASR9K Service** to add the configuration in the system.

Table 63: RADIUS AAA Proxy Settings

Parameter	Description
Display Name Priority	Order of preference for display configurations.
Accept Access Template	Structure for accepted access requests.
Proxy Access Accept Values	Criteria for accepting access requests.
Avp Subscription (List)	List of subscribed attribute-value pairs.
Additional Avps (List)	Extra attribute-value pairs for requests.
Multi Co A Template	Supports multiple Change of Authorization requests.



CHAPTER 19

Control Center

- Feature History, on page 373
- Overview, on page 373
- Functionalities, on page 374
- Login to the Control Center, on page 374
- Find Subscriber, on page 374
- Create Subscriber, on page 375
- Find Subscriber Session, on page 375
- Delete Subscriber, on page 376
- Delete Session, on page 376
- Service wise user level bifurcation, on page 377
- Configure the subscriber bifurcation report, on page 377
- Workflow, on page 378
- Subscriber search pagination in Control Center, on page 379

Feature History

Feature Name	Release Information	Description
Access Control Center from cnAAA	2025.01.0	cnAAA allows access to the Control Center through a GUI using HTTPS in a web browser. Users can view, edit, import, and export Custom Reference Data (CRD) to maintain consistent data management across systems.

Overview

The Cloud Native Authentication, Authorization, and Accounting (cnAAA) enables secure access to the Control Center through a GUI using the HTTPS protocol on a web browser.

In the **Control Center**, administrators can manage Custom Reference Data (CRD) by viewing, editing, importing, and exporting data to maintain consistency across systems. This approach is essential for preventing

discrepancies, errors, and data integrity issues that can occur when data is managed independently in multiple systems.

Functionalities

This section describes the functionalities of the Control Center.

- Login to the Control Center.
- Find Subscriber.
- Create Subscriber.
- Reference Data.
- Find Subscriber Session.

Login to the Control Center

You can login to the Control Center GUI through the https protocol.

Procedure

Enter the credentials in the **Username** and **Password** fields, of the Control Center GUI and click **Login**.

Find Subscriber

You can find detailed information of a specific subscriber using their credentials or name.

Procedure

Step 1 Enter the subscriber's name or credentials in the **Credentials** or **Name** field.

Step 2 Click **Search** to initiate the search process.

Step 3 View the results displayed in a table format.



Note Balance-Related Panels and Grids have been removed from the Subscriber Profile view.

Create Subscriber

You can create a subscriber from the control center GUI

Procedure

Step 1 In the **Credential** field, enter a unique identifier for the subscriber .This identifier is essential for tracking and managing subscriber accounts.

Step 2 In the **Name** field, enter the subscriber's full name or an alternative identifier to help with searching and managing records.

Click **Reset** to reset all the data.

Click **Save** to submit the information and complete creation of the subscriber.

Click **Save & Continue** to proceed to additional configuration screens, such as General, Credentials, Services, Notifications, and Subaccounts.

Once you click **Save & Continue**, sample data will be displayed on the Subscribers screen.

Reference Data

Step 3 **Access Reference Data:** Click Reference Data to view options for importing or exporting CRD data.

Step 4 **Manage CRD Tables:** Select the configured CRD tables to create, read, update, or delete records from the CRD tables.

Find Subscriber Session

The Find Subscriber Session allows you to search and retrieve session details for a specific subscriber based on their session data.

Before you begin

Follow these steps to configure the cnAAA Ops-Center to fetch specific subscriber session data under the Find Subscriber Session view.

1. Login to cnAAA Ops-Center.
2. Ensure the Ops-Center is fully shut down before proceeding with configuration changes.
3. Configure this command in Ops-Center.

```
cdl datastore session find-by-nuk-prefixes [ FramedIpKey MacAddressKey USuMSubscriberIdKey
 UserIdKey FramedIPv6AddressKey ]
```

4. Restart the cnAAA Ops-Center.

Procedure

Step 1 Select the **Query Key** from the drop-down list to choose the type of key for your search, such as User ID, Framed IP, or Framed IPv6 Prefix.

Step 2 In the **Key Data** field, enter the specific value associated with the selected Query Key.

Step 3 Execute the search to retrieve session details.

Review the results displayed in a structured table format, which includes the Primary Key, Type, and Subscriber Credential.

View session details by navigating to the session details section to access in-depth information about a selected session.

Delete Subscriber

You can delete a specific subscriber from Control Center GUI.

Procedure

Step 1 Navigate to **Find subscriber** and enter the subscriber's credentials or name in the **Credentials** or **Name field**.

Step 2 Click **Search** to initiate the search process.

View the results displayed in a table format.

Step 3 Place the cursor on a particular subscriber and click **Delete**.

A pop message is displayed stating "Are you sure want to delete this subscriber", click **Yes**.

You can see a **Subscriber deletion successful** message.

Delete Session

You can delete a specific subscriber session from Control Center GUI:

Procedure

Step 1 Click **Find Subscriber Session**.

Step 2 Choose the **Query Key** from the drop-down list to choose the type of key for your search, such as User ID, Framed IP, Framed IPv6 Prefix.

Step 3 In the **Key Data** field, enter the specific value associated with the selected Query Key.

Step 4 Execute the search to retrieve session details and click **Open** from table where session details are present.

Step 5 Under the session details section, click **Remove Session**.

A pop message is displayed stating "Are you sure want to remove this session", click **Yes**.

You can see a **Session deletion successful** message

Service wise user level bifurcation

The subscriber bifurcation report provides visibility into subscriber provisioning and distribution trends within the Control Center. This feature centralizes reporting, which eliminates manual database queries and reduces operational overhead. Administrators can generate, view, and export reports using filters such as circle code, service type, and daily creation counts. The system provides these reports in Comma-Separated Values (CSV) format to support business intelligence, regulatory compliance, and the analysis of baseline service assignments, such as Plan0, for new subscribers.

Software upgrade and deployment

Before you configure or use this feature, verify that the software is correctly deployed:

- **Qualify the software upgrade**: Refer to *Chapter 6: Software Upgrade Qualification* for prerequisites and system health checks.
- **Monitor and verify the upgrade**: Refer to *Chapter 6: Monitor the cnAAA Upgrade* and *Verify the Helm Status* to ensure the deployment is successful.
- **Verify post-deployment**: Refer to *Chapter 6: Post deployment verification steps* to confirm that the **Subscriber Bifurcation** link is visible in the **Control Center**.

Configure the subscriber bifurcation report

Use this procedure to configure the subscriber bifurcation report settings in the Ops-Center. These settings allow you to manage service codes, user interface (UI) display limits, and export capabilities.

This table summarizes the configurable engine properties.

Table 64: Configuration parameters

Parameter	Engine property	Default value
Plan0 service code	`com.broadhop.subscriber.bifurcation.serviceCode`	`Plan0`
UI result limit	`com.broadhop.service.bifurcation.maxResult`	1,000
Export result limit	`com.broadhop.export.service.bifurcation.maxResult`	100,000
Feature toggle	`com.broadhop.subscriber.bifurcation.enable`	`true`

Configure report settings

Procedure

Complete these steps to configure the report settings:

Property: `com.broadhop.subscriber.bifurcation.serviceCode`

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

Step 2 Navigate to your engine group: `engine <engine-group-name>`.

Step 3 Configure the Plan0 service code.

```
properties com.broadhop.subscriber.bifurcation.serviceCode value
        <service-name>
```

Step 4 Configure the UI result limit.

```
properties com.broadhop.service.bifurcation.maxResult value <number-of-results>
```

If you set this value to 0 or less, the system uses a default value of 1,000, while values greater than 100,000 are capped at 100,000. This total limit applies to all Subscriber Profile Repository (SPR) MongoDB instances combined. We recommend setting the value to 20,000 to achieve optimal UI performance.

Step 5 Configure the CSV export limit.

CSV exports cannot exceed 100,000 records. If you configure a higher value, the system adjusts it to 100,000.

```
properties com.broadhop.export.service.bifurcation.maxResult value <number-of-records>
```

Step 6 Configure query timeouts.

To prevent performance degradation during large data scans, configure query timeouts. The valid range is 1 to 60 seconds, with a default value of 20 seconds.

```
properties spr.subscriber.bifurcation.aggregate.max.time.seconds value <timeout-in-seconds>
properties spr.avp.aggregate.max.time.seconds value <timeout-in-seconds>
```

Step 7 `commit` the changes.

Workflow

Once the Subscriber Bifurcation is integrated and configured, you can generate reports by following this workflow:

1. Navigate to the **Subscribers** section in the **Control Center** and click **Subscriber Bifurcation**.
2. Select the **Subscriber Type**: **Static** (all provisioned subscribers) or **Plan0** (subscribers with the Plan0 service).
3. Apply filters by selecting the **Filter Type** (All, Provisioned, Circle Code, Service Type, or Service Code).
4. Specify the **Start Date** and **End Date**.

Note The range is limited to the previous 90 days from the current date.

5. Click **Search** to view the results in the UI.
6. Click **Export** to download the data as a CSV file.

Data interpretation and limitations

Observe these operational constraints when you use the subscriber bifurcation report:

- **Historical data constraint**: The UI is limited to a 90-day window to ensure database performance. For data older than 90 days, administrators must perform a manual database dump via the backend.
- **Time zone synchronization**: The system processes data in UTC. If your regulatory requirements mandate Indian Standard Time (IST), ensure your data team applies the necessary offset (+5:30) during external post-processing of the exported CSV.
- **Result set caps**: If the report exceeds the configured `maxResult` (default 1000), the UI truncates the display. Narrow the date range or apply more specific filters if this occurs.
- **Subscriber identification**: The report displays Network or Credential IDs. If the subscriber's full name is required, cross-reference the exported IDs using the **Find Subscriber** module.
- **Credential duplication**: If a subscriber has multiple credential IDs, the report displays them as separate rows. This ensures all active credentials are accounted for in the bifurcation.

Subscriber search pagination in Control Center

The subscriber search functionality in the Control Center supports server-side pagination to improve performance and user experience when managing large result sets. This enables efficient navigation through subscriber lists by loading manageable increments of data rather than retrieving the entire database at one time.

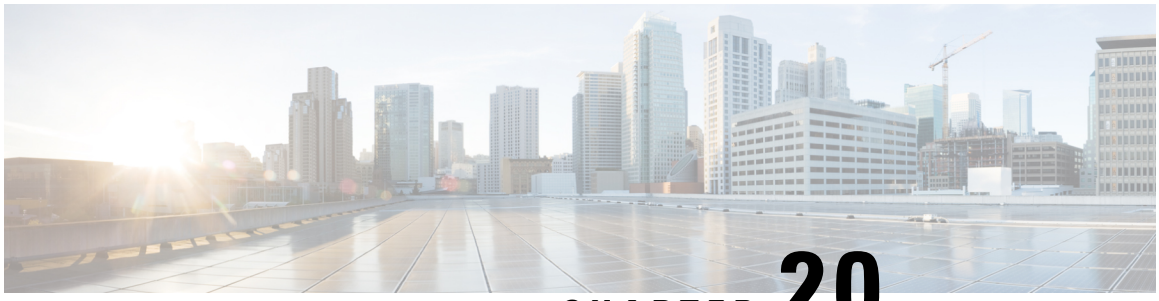
User interface enhancements

- **Page size selector**: A dropdown list that allows you to define the number of items displayed per page (10, 50, or 100 items).
- **Paging toolbar**: Navigation controls (First, Previous, Next, and Last buttons) that enable movement between result pages.
- **Loading indicators**: A loading mask that appears during search operations to provide visual feedback.
- **Status bar**: A dynamic status bar that shows the current page, the total number of results, and the pagination status.
- **Search persistence**: The system maintains search criteria across page navigation so you do not need to re-enter parameters.
- **Clear functionality**: The **Clear** link resets both the grid data and the pagination toolbar state.

The pagination uses a client-server architecture to optimize data retrieval.

Table 65: Backend and UI integration

Component	Implementation detail
Backend logic	The `SubscriberService` uses the `findSubscriberPaginated()` method, which accepts search criteria and a `PagingLoadConfig` object (offset and limit parameters).
Data retrieval	The system retrieves search results and applies pagination logic, including bounds validation, to return only the requested page of data.
Result handling	The service returns a `BasePagingLoadResult`, which includes the requested page of results and the total record count for toolbar calculation.
RPC integration	The `SubscriberService` interface and `RpcProxy` are updated to support the new paginated method signature and ensure GWT RPC compatibility.
Load listeners	Handlers manage loading states to ensure the UI remains responsive during data retrieval.



CHAPTER 20

Troubleshooting Information

- Feature Summary and Revision History, on page 381
- Debugging the cnAAA Deployment Issues, on page 382
- Issue with Refreshing the cnAAA Ops-Center, on page 388
- Subscriber Not Found or Primary Key Not Found, on page 389
- Forwarding logs to the Splunk Server, on page 391
- Centralized PCAP collection and merging, on page 393
- System health diagnostics, on page 395
- Troubleshoot Kubernetes pods during system upgrade, on page 400
- Resolve a Unified API access error, on page 404

Feature Summary and Revision History

Summary Data

Table 66: Summary Data

Applicable Product(s) or Functional Area	SMI
Applicable Platform(s)	CPC
Feature Default Setting	Enabled – Configuration required to disable
Related Documentation	Not Applicable

Feature History

Table 67: Feature History

Feature Details	Release
First introduced.	2025.01.0

Debugging the cnAAA Deployment Issues

This section describes how to debug the issues that may occur when you deploy cnAAA 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 68: 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. For information on reinstalling the charts, see Issue Refreshing the CPC Ops Center.
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 cnAAA are open in the firewall.	Use the following command to open the ports: **`firewall-cmd –zone=public –add-port=`** *`port`***`/tcp –permanent`**
Confirm if cnAAA 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 cnAAA-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`**

Auditd and sysstat service verification

This feature provides a structured approach to verify the `auditd` and `sysstat` services on Ubuntu servers. It ensures compliance with security standards, and provides system resource monitoring. It achieves this by executing standard Linux commands to check installation, service status, configurations, and log or data generation. This offers crucial insights into system security and health.

How auditd and sysstat verification works

The feature uses common system admin tools to interact with and inspect the `auditd` and `sysstat` services.

1. **Auditd service verification**:The `auditd` daemon logs security information by capturing system calls, file access, and other events based on a set of rules. `auditd` listens for events from the Linux kernel running on all the nodes, processes them by defined rules, and writes them to log files, typically in `/var/log/audit/audit.log`.

   Verification mechanism:

   - `dpkg -l | grep auditd`: Confirms `auditd` package installation.
   - `sudo systemctl status auditd`: Checks the `auditd` service status. A healthy output shows `Active: active (running)`.
   - `sudo auditctl -l`: Lists the loaded `auditd` rules. Review these rules to verify monitoring of critical security events.
   - `sudo ls -la /var/log/audit/audit.log` and `sudo stat /var/log/audit/audit.log`: These commands check the main audit log file's existence, permissions, size, and last modification time. This ensures log generation and accessibility.
   - `sudo cat /etc/audit/rules.d/audit.rules`, `sudo cat /etc/audit/audit.rules`, `sudo cat /etc/audit/rules.d/audit.rules`: Examine these files to understand the daemon's configuration and static rules.
   - `sudo systemctl enable auditd`: To enable auditd service.
   - `sudo systemctl disable auditd`: To disable auditd service.
   - `sudo systemctl stop auditd`: To stop auditd service.
   - `sudo systemctl start auditd`: To start auditd service.
   - `sudo systemctl restart auditd`: To restart auditd service.
   - `sudo auditctl -l`: to check audit `rules sudo cat /etc/audit/rules.d/audit.rules`
   - `sudo ls -la /var/log/audit/audit.log`: To check `audit logs sudo stat /var/log/audit/audit.log`

   Sample output:

```
cloud-user@gamma-master-1:~$ dpkg -l | grep auditd
ii  auditd                              1:3.1.2-2.1build1.1                     amd64
       User space tools for security auditing

cloud-user@gamma-master-1:~$ sudo systemctl status auditd
 auditd.service - Security Auditing Service
     Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled; preset: enabled)
```

```
     Active: active (running) since Tue 2025-10-14 08:11:04 UTC; 1 week 1 day ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
   Main PID: 1991 (auditd)
      Tasks: 2 (limit: 618628)
     Memory: 1022.4M (peak: 1.0G)
        CPU: 4min 49.926s
     CGroup: /system.slice/auditd.service
             └─1991 /sbin/auditd

Oct 22 05:24:49 gamma-master-1 auditd[1991]: Audit daemon rotating log files

cloud-user@gamma-master-1:~$ sudo systemctl disable auditd
Synchronizing state of auditd.service with SysV service script with
/usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable auditd

cloud-user@gamma-master-1:~$ sudo systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with
/usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable auditd (edited)
```

Note The audit system is in immutable mode. Rule changes are not allowed.

2. **Sysstat service verification**:The `sysstat` package provides tools like `sar`, `iostat`, `mpstat`, and `pidstat` for collecting, reporting, and saving system activity information. `sysstat` periodically collects system resource using `sadc` and stores it in binary data files (e.g., `saXX`) in `/var/log/sysstat/`. Tools like `sar` can then analyze these data files.

   Verification Mechanisms:

   - `dpkg -l | grep sysstat`: Confirms `sysstat` package installation.
   - `sudo systemctl status sysstat`: This command verifies the status of the sysstat service.
   - `grep -i "ENABLED" /etc/default/sysstat`: Checks the configuration flag that determines if `sysstat` data collection is active. It should show `ENABLED="true"`.
   - `systemctl list-timers sysstat-collect.timer` or `cat /etc/cron.d/sysstat`: These commands determine the frequency of system resource collection. Verifying this ensures data collection at the desired frequency.
   - `ls -la /var/log/sysstat/`: Lists the directory where `sysstat` stores collected data. The presence and recent modification times of `saXX` files confirm active data collection.
   - `sar -u`, `sar -r`, `sar -b`, `sar -n DEV`: These commands use the `sar` utility to display collected performance statistics. Successful execution confirms data collection and accessibility for reporting.
   - `iostat`, `mpstat`, `pidstat`: Test other `sysstat` tools to confirm their availability and ability to retrieve system metrics.
   - `cat /etc/sysstat/sysstat` and `cat /etc/default/sysstat`: Examine these files for `sysstat`'s overall configuration, such as log retention and the `ENABLED` flag.
   - `df -h /var/log/sysstat/`: Checks available disk space in the log directory.
   - `sudo systemctl enable sysstat`: Enables sysstatservice.

- `sudo systemctl disable sysstat`: Disables sysstat service.
- `sudo systemctl stop sysstat`: Stops sysstat service.
- `sudo systemctl start sysstat`: Starts sysstat service.
- `sudo systemctl restart sysstat`: Restarts sysstat service.
- `sudo sar`: Checks current date logs.

This section provides sample outputs for some of the CLI commands used in the verification steps:

```
admin@host:~$ dpkg -l | grep sysstat
ii  sysstat                              12.6.1-2                                amd64
    system performance tools for Linux
```

```
admin@host:~$ systemctl status sysstat
sysstat.service - Resets System Activity Logs
     Loaded: loaded (/usr/lib/systemd/system/sysstat.service; enabled; preset: enabled)
     Active: active (exited) since Fri 2025-10-28 04:20:52 UTC; 2h 8min ago
       Docs: man:sa1(8)
             man:sadc(8)
             man:sar(1)
   Main PID: 759879 (code=exited, status=0/SUCCESS)
        CPU: 7ms
```

```
admin@host:~$ sudo systemctl disable sysstat
Synchronizing state of sysstat.service with SysV service script with
/usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable sysstat
Removed "/etc/systemd/system/multi-user.target.wants/sysstat.service".
Removed "/etc/systemd/system/sysstat.service.wants/sysstat-summary.timer".
Removed "/etc/systemd/system/sysstat.service.wants/sysstat-collect.timer".
```

SNMP test trap generation facility

The SNMP test trap generation facility provides a method to simulate SNMP traps for testing and validation. This facility allows generation of SNMP traps without an actual fault or event occurring in the system. It is important to verify that monitoring and alerting infrastructure, such as Network Management Systems (NMS), receive and interpret SNMP alerts. The facility ensures that monitoring tools accurately display both "Firing" (active) and "Cleared" (resolved) alert states. This capability is essential for integration testing, troubleshooting, and User Acceptance Testing (UAT).

The facility uses a standalone script to send these simulated traps to a specified SNMP destination server. It supports generating both predefined and custom alerts, making it a flexible tool for validating monitoring system behavior.

SNMP test trap generation script

The SNMP test trap generation is performed using the `send_snmp_trap_unified.sh` script, typically located at `pcf-utilities-0` pod under `data/utilities/support/script/`.

Script Syntax:

The script supports two primary usage patterns:

- **To send all predefined alarms:**

  `./send_snmp_trap_unified.sh <TRAP_SERVER>`

- **To send a custom alarm**

  ```
  ./send_snmp_trap_unified.sh <TRAP_SERVER> <Alert_name> <Severity> <Type> <Summary>
  ```

Parameters:

- `<TRAP_SERVER>`: The IP address or hostname of the target SNMP destination server.
- `<Alert_name>`: A unique name for the custom alert (e.g., myAlert, disk-util-high).
- `<Severity>`: The severity level of the alert (e.g., critical, warning, minor).
- `<Type>`: The category or type of the alert (e.g., 'Equipment Alarm', 'Environmental'). Enclose values with spaces in single quotes.
- `<Summary>`: A brief description or message for the alert (e.g., `'Disk usage high'`, `'CPU temperature exceeded'`). Enclose values with spaces in single quotes.

To display the script's usage information, execute the script with the `--help` or `-h` option:

```
./send_snmp_trap_unified.sh --help
```

How the script generates test traps

When the `send_snmp_trap_unified.sh` script executes, it simulates a complete alert lifecycle for each trap it sends. The process involves three distinct steps for each alert:

1. **Sends Firing Trap:** The script first dispatches an SNMP trap indicating a "Firing" (active) state for the specified alert.
2. **Waits:** After sending the firing trap, the script pauses for five seconds. This delay simulates the time an alert might remain active before being resolved.
3. **Sends Clearing Trap:** Following the waiting period, the script dispatches a second SNMP trap indicating a "Cleared" (resolved) state for the same alert.

Note This sequence applies to both predefined alarms and custom alarms, ensuring that monitoring systems can be tested for their ability to handle both alert initiation and resolution.

Generate predefined SNMP test traps

Perform these steps to send a custom SNMP test trap to a target server.

Procedure

Step 1 Access the server where the `send_snmp_trap_unified.sh` script resides. The typical path is `/data/utilities/support/script/`.

Step 2 Execute the script, providing the IP address of the target SNMP server.

```
./send_snmp_trap_unified.sh 192.0.2.1
```

Step 3 The script sends a series of predefined firing and clearing traps. Output similar to the following appears:

```
=== Sending All Predefined Alarms ===
Total alarms to send: 24
Target server: 192.0.2.1

=== Sending Alarm ===
Alert: cpu-util-idle
Severity: critical

Sending firing trap...
SNMP firing trap sent successfully!
 Firing alarm 'cpu-util-idle' sent successfully!
Waiting 5 seconds before sending clearing trap...

=== Sending Clearing Trap ===
Sending clearing trap...
SNMP clearing trap sent successfully!
 Clearing trap for 'cpu-util-idle' sent successfully!
```

Generate custom SNMP test traps

Perform these steps to send a custom SNMP test trap to a target server.

Procedure

Step 1 Access the server where the `send_snmp_trap_unified.sh` script resides. The typical path is `/data/utilities/support/script/`.

Step 2 Execute the script, providing the IP address of the target SNMP server along with the custom alert details.

```
./send_snmp_trap_unified.sh 192.0.2.1 myAlert critical 'Equipment Alarm' 'Disk usage high'
```

Step 3 The script sends a firing trap followed by a clearing trap for the custom alert. Output similar to the following appears:

```
Sending firing trap...
SNMP firing trap sent successfully!
 SNMP firing trap sent successfully!
Waiting 5 seconds before sending clearing trap...

=== Sending Clearing Trap ===
Sending clearing trap...
SNMP clearing trap sent successfully!
 SNMP clearing trap sent successfully!
```

Verify trap reception and interpretation

After generating SNMP test traps, verify the monitoring system's behavior by observing the target SNMP server or NMS. Adhere to these principles for effective verification.

- **Confirm trap reception**: Ensure the SNMP server successfully receives both the "Firing" and "Cleared" traps.
- **Validate field display**: Verify that all alert fields provided to the script (Alert Name, Severity, Type, Summary) are correctly displayed in the monitoring console.

- **Differentiate alert states**: Confirm that the monitoring system accurately differentiates between "Firing" and "Cleared" states for the same alert. The system should show the alert becoming active and then resolving.
- **Check IPv4/IPv6 support**: If applicable, verify that the integration works correctly for both IPv4 and IPv6 communication.

Issue with Refreshing the cnAAA Ops-Center

This section describes how to refresh the cnAAA Ops-Center to display the latest configurations.

Issue

The cnAAA Ops-Center is not considering the recent configurations due to which you may observe stale data or not get the expected response.

Solution

You can refresh the cnAAA Ops-Center using the basic and advanced steps. Perform the advanced steps only when the basic steps do not resolve the issue.

Basic Steps

1. Run this command to undeploy cnAAA from the Ops-Center:

   system mode shutdown

2. Use the following to manually purge any pending deployments from the helm:

   helm delete --purge *helm_chart_name*

3. From the master node, run the following to delete the configMaps from the namespace where cnAAA is installed:

   kubectl delete cm *config_map_name* **-n** *namespace*

4. Run this command to 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 this command to delete the configMap:

      Kubectl delete configmap *configmap_name* **-n** *namespace*

5. Use these commands to reinstall the helm chart. Once the chart is installed, a new instance of the cnAAA Ops-Center is available.

   helm upgrade –install *release name addR/chart_name* **-f** *filenames* **--namespace** *namespace*

Advanced Steps

1. Remove the cnee-ops-center.
2. Delete the configMaps from the namespace.

   For more information on step 1 and 2, see the **Basic** steps.
3. Install the cnAAA Ops-Center.

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.

Use the following command to view the pod's logs:

kubectl describe pod *pod_name* **-n** *namespace*

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	Then
Status column displays the state as Running, and the Ready column has the same number of containers on both sides of the forward-slash (/)	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.
Status column displays the state as Pending, Waiting, or CrashLoopBackOff	Run the following to review the details such as the messages, reasons, and other relevant information: **kubectl describe pod** *pod_name* **-n** *namespace*
Status is init or ContainerCreating	Pod is in the process of starting up.
Status is Running, and in the Ready column the number of containers on both sides of forward-slash (/) are different	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.
Status and Ready columns, and logs of the container do not indicate any issue	Verify that the required httproutes or the service that is required to reach the application is up and running.

Subscriber Not Found or Primary Key Not Found

This section describes how to resolve the issues that report the Subscriber Not Found or Primary Key Not Found messages.

Problem

When the NFs cannot find the subscriber details, they send the Subscriber Not Found or Primary Key Not Found to cnAAA.

Resolution

1. Analyse the logs of the cnAAA Engine and RADIUS endpoint pod for the subscriber or primary key related issues.

   On the master node, run the following command to determine the engine and radius-ep pod.

   kubectl logs -n *namespace pod_name*

2. Navigate to the pods and review the subscriber availability status and the subscriber count in the database. Based on the subscriber's status, take the appropriate action to resolve the issue.

   cdl show session *count/summary*

Called station id report generation

The proposed script first retrieves session information from the ops-center CLI using the command cdl show sessions summary. Each session includes non-unique keys, including the Called-Station-Id variable. The script then compares the Called-Station-Id for each session with the corresponding provisioned Called-Station-Id by checking the subscriber details.

Logging Called-Station-Id in report:

This section describes when to log the called-station-id in the comma-separated values (CSV) log file.

- **Log the called-station-id in the CSV log file if**:
  - The provisioned and session called-station-id are different.
  - Only the session called-station-id exists (the provisioned value is absent).
- **Do not log the called-station-id if**:
  - The provisioned and session called-station-id match.
  - Only the provisioned called-station-id exists (the session value is absent).

1. Provisioned = “A”, Session = “A”

   Do not log.

2. Provisioned = “A”, Session = “B”

   Log.

3. Provisioned = (none), Session = “A”

   Log.

4. Provisioned = “A”, Session = (none)

   Do not log.

Generate a called station report

Follow these steps to generate a called station report:

Procedure

Step 1 Install the latest cnAAA build in the setup.

Step 2 Verify that all pods are running at one hundred percent.

Step 3 Check that the *pcf-utilities-0* pod is running at one hundred percent.

Step 4 Run these command to confirm the pod status:

```
kubectl get pod -n pcf | grep pcf-utilities-0
pcf-utilities-0 1/1 Running 0 2d1h
```

Step 5 Log in to the *pcf-utilities-0* pod and navigate to the script folder: `data/utilities/support/script`

Step 6 Copy the **called_stn_id_report_gen.py** script to the master node.

Note You cannot execute the script from a pod. The script requires `kubectl` to connect to MongoDB pods and retrieve subscriber information, but `kubectl` is not installed within the pod.

Step 7 Run the Radius call flow with the called station ID.

Step 8 Execute the Python script using these command:

```
python3 called_stn_id_report_gen.py --namespace <value> --replicaname <replica set names> --spr_dbs
 <IP:port of replica set>
```

Example:

```
python3 called_stn_id_report_gen.py --namespace pcf --replicaname sdb-subscriber2,sdb-subscriber3
--spr_dbs 10.1.43.70:65002,10.1.43.70:65003
```

Note You can pass multiple replica sets as arguments in this script.

Step 9 Verify the generated CSV report.

Note If you enable the volume (`vol`), the system attaches the mount to the node. You can then execute the script directly from the node. The script is available at `/data/pcf/data-pcf-utilities-0/support/script`.

Forwarding logs to the Splunk Server

This section describes how to enable cnAAA to forward the logs to the Splunk server.

Splunk is a third-party monitoring application that stores the log files and provides index-based search capability. You can configure cnAAA 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.

Use the following configuration to forward the logs to the Splunk server.

```
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
```

The following 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.

Centralized PCAP collection and merging

Feature Name	Release Information	Description
Centralized PCAP collection and merging	2026.02.0	The Mergecap utility simplifies troubleshooting. It does this by remotely executing `tcpdump` across multiple cluster nodes in parallel, then automatically merging the resulting packet capture files into a single trace.

The Mergecap utility is an on-demand, Linux CLI based troubleshooting tool. It can trigger packet captures on multiple cluster nodes simultaneously from the `pcf-utilities` pod. The utility then automatically merges the resulting Packet Capture (PCAP) files into one consolidated file. This feature eliminates manual PCAP collection and correlation. It minimizes timestamp skew across nodes and ensures zero impact on the system memory footprint when not in use.

Prerequisites

Before start using the utility, ensure that you meet this requirement:

- **Passwordless SSH**: Ensure all Kubernetes cluster nodes are accessible over SSH from any master node without a password.

Note Replace `cloud-user` with your actual SSH username and `cpc` with your actual namespace throughout all commands in this guide.

SSH Key Setup (Optional)

If passwordless SSH is not configured, generate an SSH key and copy it to all nodes:

1. Generate SSH Key (on any master node): `ssh-keygen -C ""`
2. Copy SSH Key to All Cluster Nodes:

```
for node in $(kubectl get nodes --no-headers | awk '{print $1}'); do
    echo "Copying ssh key into $node"
    ssh-copy-id cloud-user@$node
done
```

Note When prompted, enter the password for the nodes.

Configure and run the packet capture utility

Use this procedure to capture network traffic across multiple cluster nodes in parallel and merge the resulting files into a single trace.

Procedure

Step 1 Copy SSH Key to the `pcf-utilities` pod to enable passwordless SSH connections to individual nodes:

```
kubectl cp -n cpc /home/cloud-user/.ssh/id_ed25519
pcf-utilities-0:/data/utilities/support/script/pcap_merge/
```

Step 2 Configure the script variables:

a. Log in to the utility pod:

```
kubectl exec -it -n cpc pcf-utilities-0 -- bash
```

b. Navigate to the script directory:

```
cd /data/utilities/support/script/pcap_merge
```

c. Open the script for editing:

```
vi pcap_merge.sh
```

Step 3 Update the variables for your environment.

Table 69: Script Variables

Variable	Default Value
`USERNAME`	cloud-user
`SSH_KEY`	$SCRIPT_DIR/id_ed25519
`INTERFACE`	any
`PORTS`	(port 1812 or port 1813)
`TIMEOUT`	180 seconds
`OUTPUT_DIR`	$SCRIPT_DIR/captures
`NODES`	All k8s nodes

Step 4 Run the capture script.

```
./pcap_merge.sh
```



Note The script collects `tcpdump` data for the configured `TIMEOUT`. To stop the collection before the configured TIMEOUT, press **Ctrl+C**. The script catches the interrupt signal and proceeds with the merge operation.

Step 5 Navigate to the output directory (default: ./captures) and ensure the following files exist:

- **Per-node PCAP**: `radius_<NODE_IP>_<TIMESTAMP>.pcap`

- **Merged PCAP**: `all_radius_<TIMESTAMP>.pcap`
- **Cleanup**: Manually delete any old PCAP files that are no longer needed.

Step 6 (Optional) Verify that the sum of packets captured in the individual files matches the total in the merged file:

```
cd /data/utilities/support/script/pcap_merge/captures/
for pcap in $(ls radius*); do
    echo "Number of packets captured in : $pcap"
    capinfos -Mc $pcap | grep "Number" | cut -d ":" -f 2 | tr -d " "
done
echo "Number of packets captured in Merged file : "
capinfos -Mc all_radius*.pcap | grep "Number" | cut -d ":" -f 2 | tr -d " "
```

System health diagnostics

Feature overview

The `cpc-system cnaaa-diagnostics all` command provides a consolidated health overview of the cnAAA environment. This command aggregates critical data into a single, structured report, which eliminates the need to execute multiple Kubernetes commands manually. The report consolidates health status for the following areas:

- **Deployment topology**: High availability (HA) or geo-redundant (GR) environment status.
- **Infrastructure health**: Pod status across cnAAA namespaces, Kubernetes node readiness, Persistent Volume (PV), Persistent Volume Claim (PVC) status, and service availability.
- **Resource metrics**: Disk utilization for nodes, and clock synchronization.
- **Application health**: Engine and RADIUS application status, LBS service health, and MongoDB replica set status.
- **Synchronization and licensing**: Geo-redundancy status, and Smart License status.
- **Active alerts**: Displays the alerts which are active in the system along with the severity.

Run the system health diagnostics

Before you begin

Before you run the diagnostics command, ensure that you meet these requirements:

- **Passwordless SSH**: For the system to return data for node disk usage, active alerts, clock synchronization, PV/PVC status, and Radius LBS status, configure passwordless SSH communication between the `pcf-utilities-0` pod and all cluster nodes:
  - Generate key pair locally:

    ```
    ssh-keygen -C ""
    ```

  - Copy public key to nodes:

```
for node in $(kubectl get nodes --no-headers | awk '{print $1}'); do
ssh-copy-id <user-name>@$node
done
```

- Copy private key and username into utilities pod and set permissions:

```
kubectl <pcf-namespace> cp ~/.ssh/id_ed25519
pcf-utilities-0:/data/utilities/id_ed25519
kubectl -n <pcf-namespace> pcf-utilities-0 -- chmod 600 /data/utilities/id_ed25519
echo -n <username> | kubectl <pcf-namespace> exec -i pcf-utilities-0 -- tee
/data/utilities/pcf_diag_ssh_user > /dev/null
```

- Persistence: To ensure the username and SSH key persist across system reboots or pod deletions and restarts, store them in `/data/utilities/`. This directory is backed by a PV, ensuring the configuration is restored automatically.

Note If you do not complete these prerequisite steps, the features dependent on SSH (Node disk usage, active alerts, clock status, PV/PVC status, and Radius LBS status) will not function. The script will print an error message stating that passwordless SSH is not enabled.

Generating diagnostic reports

Use this procedure to generate a system health report for the cnAAA environment.

Procedure

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

Step 2 Execute the `cpc-system cnaaa-diagnostics all` command.

Step 3 Verify that the report includes all expected sections, such as deployment topology, pod status, service endpoints, disk usage, and MongoDB replica set status.

Note A successful verification is indicated by the presence of the full report and the absence of `[FAIL]` or `[WARN]` tags in critical sections.

```
[gamma/gamma-cncps] pcf# cpc-system cnaaa-diagnostics all
Fri Jul  17 13:00:32.764 UTC+05:30
result
CNAAA Diagnostics GR Environment
================================
CNAAA NODE STATUS
-----------------
gamma-master-1 is ready [PASS]
gamma-master-2 is ready [PASS]
gamma-master-3 is ready [PASS]
```

```
CNAAA POD STATUS
----------------
PCF Namespace [pcf-gamma-cncps]: All pods running [PASS]
CEE Namespace [cee-gamma-cncps-cee]: All pods running [PASS]
LBS Namespace [lbs-gamma]: All pods running [PASS]

CNAAA Service Port Checks [pcf-gamma-cncps]
-------------------------------------------
All service ports reachable [[PASS]]

CNAAA Service Port Checks [cee-gamma-cncps-cee]
-----------------------------------------------
smart-agent-cee-gamma-cncps-cee-ops-center: 10.103.181.198:8888/TCP [FAIL]

CNAAA Service Port Checks [lbs-gamma]
-------------------------------------
smart-agent-lbs-gamma-ops-center: 10.103.174.41:8888/TCP [FAIL]

Disk space on Nodes
-------------------
Checking disk space on gamma-master-1 43% [PASS]
Checking disk space on gamma-master-2 39% [PASS]
Checking disk space on gamma-master-3 39% [PASS]

Clock Sync Status
-----------------
Checking clock synchronization on nodes [PASS]

CNAAA PVC & PV Status
---------------------
All PVCs are bounded [PASS]

All PVs are bounded [PASS]

Active alerts
-------------
NAME                                SEVERITY  STARTS AT       SUMMARY
-------------------------------------------------------------------------------------------------------------------------------
network-receive-errors           major     07-17T07:06:05  Network interface "bd1" showing
 receive errors on hostname gamma-master-1"
SvnRepoDeleted                  critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB3
SmartLicenseEvalModeExpire        major     07-17T04:43:13  Smart license eval expired.
SvnRepoDeleted                  critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB6
kafka-pod-down                    major     07-17T04:46:17
kafka-pod-down                    major     07-17T04:44:47
network-receive-errors            major     07-17T07:06:05  Network interface "ens1f0"
showing receive errors on hostname gamma-master-1"
watchdog                        minor     07-17T04:42:10  This is an alert meant to ensure
 that the entire alerting pipeline is functio...
SvnRepoDeleted                  critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB5
outbound_request_ProxyAccounting  critical  07-17T04:48:45  This is an alert that radius
outbound_request_ProxyAccounting is less than ex...
network-receive-errors           major     07-17T06:23:45  Network interface "bd1" showing
 receive errors on hostname gamma-master-3"
network-receive-errors            major     07-17T06:23:45  Network interface "ens1f0"
showing receive errors on hostname gamma-master-3"
grpc_message_access_send_total    critical  07-17T04:48:41  This is an alert that
grpc_message_access_send_total is less then expected
policy-engine-acct-request        critical  07-17T06:27:41  This is an alert that
policy-engine-acct-request less than expected
disk-util-high                    critical  07-17T04:43:13
```

```
jvm_memory                          critical  07-17T04:45:12  This is an alert that
jvm_memory_bytes_used is more than expected
SmartLicenseEvalModeExpire          major     07-17T04:47:23  Smart license eval expired.
SmartLicenseEvalModeExpire          major     07-17T04:47:23  Smart license eval expired.
disk-util-high                      critical  07-17T04:43:13
SvnRepoDeleted                     critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB2
kafka-pod-down                      major     07-17T04:46:57
SmartLicenseEvalModeExpire          major     07-17T04:47:23  Smart license eval expired.
disk-util-high                      critical  07-17T04:43:13
SvnRepoDeleted                     critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB1
SvnRepoDeleted                     critical  07-17T04:45:52   Expected SVN Repository missing
 is: TestPB4
policy-engine-access-request       critical  07-17T04:48:41  This is an alert that
policy-engine-access-request less than expected

Geo Sync Status
---------------
Pod: cdl-ep-session-c1-d0-77949d78-g7scn
2026/07/17 13:00:58 Geo sync is successful

Pod: cdl-ep-session-c1-d0-77949d78-vmsn2
2026/07/17 13:01:00 Geo sync is successful

Smart Agent Status
------------------
Smart agent is enabled [PASS]

Radius LBS Status
-----------------
Radius LBS Configuration is enabled [PASS]

Checking on Radius EMP Service
        Radius EMP Service (v4, v6) [PASS]

Checking on Radius LB Service
        Radius LB Service (v4, v6) [PASS]

Checking on Radius CRD Status
        Required CRDs [PASS]

Checking on Radius ETCD key values [PASS]

Verifying iptables/ip6tables NAT rules on radius nodes
        Node: gamma-master-1
            IPv4: (DNATs=3, SNATs=1) [PASS]
            IPv6: (DNATs=3, SNATs=1) [PASS]
        Node: gamma-master-2
            IPv4: (DNATs=3, SNATs=1) [PASS]
            IPv6: (DNATs=3, SNATs=1) [PASS]
        Node: gamma-master-3
            IPv4: (DNATs=3, SNATs=1) [PASS]
            IPv6: (DNATs=3, SNATs=1) [PASS]

REPLICA SET STATUS
------------------

                                                                          PRIORITY  PRIORITY
STATE      STATE          ARBITER  ARBITER  REPLICATION  SITE
NAME       HOST         PORT   MEMBER NAME          NODE NAME       RUNNING   CONFIG
PRIMARY    MEMBER         RUNNING  CONFIG   LAG
-------------------------------------------------------------------------------------------
sdb-spr01  10.172.2.42  65001  sdb-rs1-s2-m1        delta-master-1  102       102
```

```
SECONDARY  SECONDARY      false    false    0.0          remote
sdb-spr01  10.172.2.33  65001  sdb-rs1-s1-m2        gamma-master-2  103       103
SECONDARY  SECONDARY      false    false    10.0         local
sdb-spr01  10.172.2.32  65001  sdb-rs1-s1-m1        gamma-master-1  104       104
PRIMARY    PRIMARY        false    false    0.0          local
sdb-spr01  10.172.2.22  65001  sdb-rs1-s3-arbiter3  UNKNOWN         UNKNOWN
UNKNOWN    NO_CONNECTION  false    true     UNKNOWN      remote
sdb-spr01  10.172.2.34  65001  sdb-rs1-s1-arbiter1  gamma-master-3  0
ARBITER    ARBITER        true     true     N/A          local
sdb-spr01  10.172.2.44  65001  sdb-rs1-s1-arbiter2  delta-master-3  0
ARBITER    ARBITER        true     true     N/A          remote
sdb-spr01  10.172.2.43  65001  sdb-rs1-s2-m2        delta-master-2  101       101
SECONDARY  SECONDARY      false    false    0.0          remote
sdb-spr02  10.172.2.33  65002  sdb-rs2-s1-m2        gamma-master-2  103       103
SECONDARY  SECONDARY      false    false    0.0          local
sdb-spr02  10.172.2.32  65002  sdb-rs2-s1-m1        gamma-master-1  104       104
PRIMARY    PRIMARY        false    false    0.0          local
sdb-spr02  10.172.2.42  65002  sdb-rs2-s2-m1        delta-master-1  102       102
SECONDARY  SECONDARY      false    false    0.0          remote
sdb-spr02  10.172.2.44  65002  sdb-rs2-s1-arbiter2  delta-master-3  0
ARBITER    ARBITER        true     true     N/A          remote
sdb-spr02  10.172.2.22  65002  sdb-rs2-s3-arbiter3  UNKNOWN         UNKNOWN
UNKNOWN    NO_CONNECTION  false    true     UNKNOWN      remote
sdb-spr02  10.172.2.34  65002  sdb-rs2-s1-arbiter1  gamma-master-3  0
ARBITER    ARBITER        true     true     N/A          local
sdb-spr02  10.172.2.43  65002  sdb-rs2-s2-m2        delta-master-2  101       101
SECONDARY  SECONDARY      false    false    0.0          remote
sdb-spr03  10.172.2.33  65003  sdb-rs3-s1-m2        gamma-master-2  103       103
SECONDARY  SECONDARY      false    false    0.0          local
sdb-spr03  10.172.2.34  65003  sdb-rs3-s1-arbiter1  gamma-master-3  0
ARBITER    ARBITER        true     true     N/A          local
sdb-spr03  10.172.2.32  65003  sdb-rs3-s1-m1        gamma-master-1  104       104
PRIMARY    PRIMARY        false    false    0.0          local
sdb-spr03  10.172.2.42  65003  sdb-rs3-s2-m1        delta-master-1  102       102
SECONDARY  SECONDARY      false    false    0.0          remote
sdb-spr03  10.172.2.22  65003  sdb-rs3-s3-arbiter3  UNKNOWN         UNKNOWN
UNKNOWN    NO_CONNECTION  false    true     UNKNOWN      remote
sdb-spr03  10.172.2.44  65003  sdb-rs3-s1-arbiter2  delta-master-3  0
ARBITER    ARBITER        true     true     N/A          remote
sdb-spr03  10.172.2.43  65003  sdb-rs3-s2-m2        delta-master-2  101       101
SECONDARY  SECONDARY      false    false    0.0          remote
```

Diagnostic command arguments

Use these arguments with `cpc-system cnaaa-diagnostics` to retrieve specific health reports:

Table 70: Status indicators

Argument	Description
`all`	Generates a comprehensive health report of node status, pod status, service ports, node disk usage, smart agent status, PV/PVC status, radius-lbs checks, active alerts, geo sync status, and replica set status.
`node-status`	Reports readiness status of all Kubernetes nodes. Returns `[PASS]` if all nodes are `Ready`, or `[FAIL]` if any node is `NotReady`.

Argument	Description
`pod-status`	Verifies health of pods across PCF, CEE, and LBS namespaces. Returns `[PASS]` if all pods are `Running` and ready. Returns `[FAIL]` for `Pending`, `CrashLoopBackOff`, `ImagePullBackOff`containers. Returns `[FAIL]` if the namespace is not detected.
`service-port`	Validates TCP reachability for all service endpoints. Returns `[PASS]` for successful TCP handshake, `[FAIL]` if unreachable.
`active-alerts`	Retrieves currently active alerts from the monitoring subsystem. Returns alerts if present; prints "Active Alerts: None" if healthy.
`engine-health`	Executes in-pod health checks for Engine pods. Returns `[PASS]` if successful, `[FAIL]` if the command terminates with a non-zero exit code.
`radius-health`	Executes in-pod health checks for Radius pods. Returns `[PASS]` if successful, `[FAIL]` if the command terminates with a non-zero exit code or port check fails.
`geo-sync-status`	If it is a GR sytem, it checks on the geo sync between two sites.
`replica-set`	Retrieves MongoDB replica-set health, including state, priority, and synchronization status. Returns `[PASS]` if members are healthy, `[FAIL]` if the state is unhealthy.
`pvc-pv-status`	Reports the status of PVCs and PVs. Returns `[PASS]` if all are `Bound`, `[FAIL]` for `Pending` or `Released` volumes, and reports `Volumes are not configured` if none exist.
`radius-lbs-status`	Validates LBS configuration and runs EMP, LB, CRD, ETCD-key, and iptables/ip6tables checks. Returns `[PASS]` if enabled and checks succeed, `[FAIL]` if LBS is disabled or pods are unavailable.

Troubleshoot Kubernetes pods during system upgrade

Feature overview

Containers or pods might not reach a running state during a system upgrade or startup. Use the procedures and reference information in this section to diagnose and resolve container and pod issues.

Basic troubleshooting commands:

Use these commands to view the status of the pod and the cluster:

- To assess the overall health of pods, run `kubectl get pods -n <namespace> -o wide` and examine the `READY`, `RESTARTS`, and `STATUS` columns.
- Identify pods that are not running or ready. Use the `kubectl get pods -n <namespace> | grep -Ev '([0-9]+)/\1.*Running'` to filter for non-operational pods.
- Run this command to displays events related to image pulls, probe failures, or scheduling to view image pull, probe failure, or scheduling events:

  ```
  kubectl get events -n <namespace> --sort-by=.lastTimestamp | tail
          -40
  ```

- Use this command to identify disk, memory, or PID pressure on a node:

  ```
  kubectl describe nodes | grep -A5 -iE 'Conditions|Pressure'
  ```

Pod status interpretation:

Use this table to understand the current pod status.

Table 71: Pod status descriptions

Status	Description
`2/2 Running`	The pod is healthy.
`1/2 Running`	The pod is running. However, one container did not pass its readiness probe, so it is out of service.
`0/1`, `CrashLoopBackOff`, `Error`	The system could not start the container.
`Pending`	The system is experiencing one or more of these issues: scheduling problems, Persistent Volume Claim (PVC) errors, or other resource limitations.

Troubleshoot pod startup failures:

Use this procedure to diagnose and resolve issues when pods are in a `CrashLoopBackOff`, `Error`, or `NotReady` pod states.

1. Describe the pod, then review the events section. Check for messages such as `Liveness/Readiness probe failed`, `Back-off restarting`, `OOMKilled`, `FailedScheduling`, `ErrImagePull`, or `ImagePullBackOff`.

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

2. Check the container exit reason and the last state by running this command:

   ```
   kubectl get pod -n <namespace> <pod-name> -o jsonpath='{range
   .status.containerStatuses[*]}{.name}{": "}{.state}{" /
   lastState="}{.lastState}{"\n"}{end}'
   ```

3. To view logs for a container that is failing, check the current and previous crash logs using the command:

   ```
   kubectl logs -n <namespace> <pod-name> -c <container-name>
   kubectl logs -n <namespace> <pod-name> -c <container-name> --previous
   ```

Table 72: Common failure causes

Signal in describe or logs	Likely cause
`OOMKilled`	Memory limit is too low.
`CreateContainerConfigError`	The ConfigMap or Secret reference is invalid.
`ErrImagePull` or `ImagePullBackOff`	The image is missing from the node.

Log file system path and access commands::

This table lists the log files available on the system for detailed troubleshooting.

Table 73: Log locations

Log name	Path	Description	Access command
coaBackOffFailureReport.csv	/data/consolidated-aaa-logging path of aaa logging pod	If all CoA entries for BNG fail, the system creates a single entry that contains both CoA and BNG details.	`kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/`
pmz_consolidated-retryandsuccess.csv	/data/consolidated-aaa-logging path of aaa logging pod	If a proxy request fails on the first attempt but succeeds on subsequent attempts, the system records a single entry. This entry includes the proxy and PMZ OCS details.	`kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/`
pmz_consolidated-retryandfailure.csv	/data/consolidated-aaa-logging path of aaa logging pod	If a proxy request fails on all attempts, an entry is created that records the proxy and PMZ OCS details.	`kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/`
ocs_consolidated-retryandsuccess.csv	/data/consolidated-aaa-logging path of aaa logging pod	If a proxy request fails on the first attempt but succeeds on subsequent attempts, a single entry will be created to capture the proxy and OCS details.	`kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/`
ocs_consolidated-retryandfailure.csv	/data/consolidated-aaa-logging path of aaa logging pod	If every proxy request fails, the system creates one entry that records both proxy and OCS details.	`kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/`

Log name	Path	Description	Access command
consolidated-api.log	/data/consolidated-aaa-logging path of aaa logging pod	Logs for Unified API CRUD operations.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
cpc-custrefdata_audit.log	/data/consolidated-aaa-logging path of aaa logging pod	Logs for CRD CRUD operations.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
bng-device-audit.log	/data/consolidated-aaa-logging path of aaa logging pod	Logs for BNG addition CRUD operations.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
radius-aaa-audit.log	/data/consolidated-aaa-logging/ path of aaa logging pod	Logs for Radius server addition done via script.	kubectl exec -it -n <NAMESPACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
cpc-pb-publish-svn-diff.log	POD: consolidated-aaa Path: /data/consolidated-aaa-logging for all the applicable columns	Logs for PB publish and SVN differences.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
cpc-audit-pb.log	/data/consolidated-aaa-logging path of aaa logging pod	Logs for PB operations (login, logout, save repository, and publish).	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
lastlogin_user_pb.log	/data/consolidated-aaa-logging path of aaa logging pod	Captures username and timestamp for the last PB user login.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
lastlogin_user_cc.log	/data/consolidated-aaa-logging path of aaa logging pod	Captures the username and the timestamp associated with the most recent CC user login.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/

Log name	Path	Description	Access command
cpc-audit-cc.log	/data/consolidated-aaa-logging path of aaa logging pod	Logs for CC CRUD operations, including login and logout events.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
consolidated-engine.log	/data/consolidated-aaa-logging path of aaa logging pod	Policy engine logs detailing request processing.	kubectl exec -it -n <NAMEPSACE> consolidated-aaa-logging-0 -- bash cd /data/consolidated-aaa-logging/
journalctl logs	/var/log/journal	Combination of all pod logs running on that node	journacl -f or journalctl --since "20 minutes ago"
pod logs	/var/log/pods/<pod-id>/<Container-name>	Contains all pod logs running on the node separately	kubectl exec -it -n <NAMEPSACE> <POD-NAME> -- bash
mongo logs	/var/log/pods/<mongo-pod-id>/<Container-name>	Contains all MongoDB logs related to election, replication, and CRUD operations.	kubectl exec -it -n <NAMEPSACE> <POD-NAME> -- bash

Resolve a Unified API access error

Restore access to the Unified API URL when the whitelist does not include the system IP address.

The Unified API whitelist allows access from listed IP addresses or ranges. Requests from addresses that are not listed are denied.

Before you begin

You have access to the CPC Ops-Center CLI.

Procedure

Step 1 Display the configured Unified API whitelist.

Example:

```
show running-config api unified whitelistIPs
```

Step 2 Check whether the system IP address that needs access appears in the command output.

Step 3 If the system IP address is not listed, enter configuration mode and add the required IP address or CIDR range to the whitelist. Include the other addresses or ranges that must retain access.

Example:

```
config
api unified whitelistIPs [ <system-ip-or-cidr>
```

Step 4 Commit the configuration.

Example:

```
commit
```

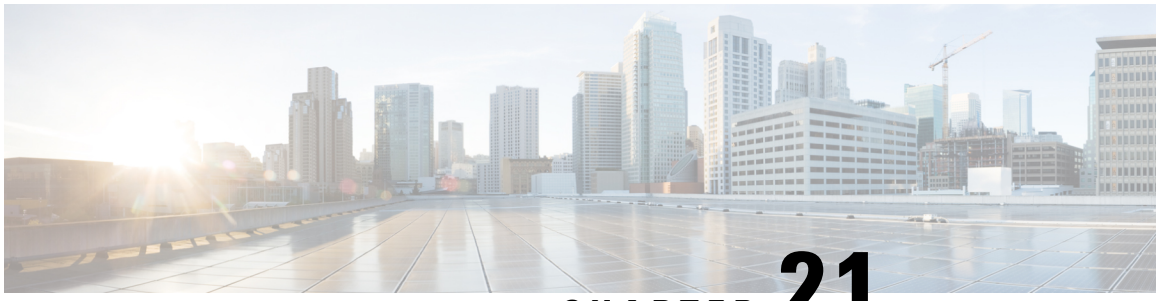
Step 5 Display the configured whitelist to verify the change.

Example:

```
show running-config api unified whitelistIPs
```

Step 6 Retry the Unified API URL.

The system IP address is included in the whitelist and can access the Unified API URL.



CHAPTER 21

Sample cnAAA Configuration

- Sample cnAAA Configuration, on page 407
- cnAAA GR configurations, on page 416
- Sample ULB Configuration, on page 452

Sample cnAAA Configuration

This configuration file is an example provided for reference purposes. You must create and modify your own configuration file according to the specific needs of your deployment. Use plain text password as applicable.

Note The mandatory parameters are required to ensure that the critical pods such as Policy Builder, CRD, DB (admin and SPR), RADIUS-End point and Policy Engine are in the running state.

```
cloud-user@alpha-master-1:~$ kubectl get pods -A
NAMESPACE          NAME                                                    READY
  STATUS    RESTARTS       AGE
cee-alpha          alert-logger-797bc6465-hdlqj                            1/1
  Running   0              7d21h
cee-alpha          alert-router-674f59995d-fzmlb                           1/1
  Running   0              7d21h
cee-alpha          alertmanager-0                                          2/2
  Running   0              7d23h
cee-alpha          alertmanager-1                                          2/2
  Running   0              7d21h
cee-alpha          alertmanager-2                                          2/2
  Running   0              7d21h
cee-alpha          alertmanager-config-sync-6f8f74cbf6-fkv72               1/1
  Running   1 (2d4h ago)   7d18h
cee-alpha          blackbox-exporter-h8nlx                                 1/1
  Running   1              13d
cee-alpha          blackbox-exporter-qdmrh                                 1/1
  Running   1              13d
cee-alpha          blackbox-exporter-xc522                                 1/1
  Running   1              13d
cee-alpha          bulk-stats-0                                            3/3
  Running   0              7d23h
cee-alpha          bulk-stats-1                                            3/3
  Running   0              7d21h
cee-alpha          cee-alpha-product-documentation-67846bdffb-8w7xs         2/2
  Running   0              7d21h
```

Sample cnAAA Configuration

```
cee-alpha         core-retriever-7ctxq                                          2/2
  Running   2             13d
cee-alpha         core-retriever-l8x2v                                          2/2
  Running   2             13d
cee-alpha         core-retriever-mpgv5                                          2/2
  Running   2             13d
cee-alpha         core-retriever-wpl76                                          2/2
  Running   2             13d
cee-alpha         core-retriever-zlxjk                                          2/2
  Running   2             13d
cee-alpha         grafana-555f44b84-7s6qc                                       2/2
  Running   0             7d21h
cee-alpha         grafana-555f44b84-wv9sr                                       2/2
  Running   0             7d21h
cee-alpha         grafana-dashboard-metrics-5dc47f98b-vcdvt                     1/1
  Running   0             7d21h
cee-alpha         kube-state-metrics-5f7fcfccf-b8j7c                            1/1
  Running   0             7d21h
cee-alpha         logs-retriever-6j2zf                                          1/1
  Running   1             13d
cee-alpha         logs-retriever-9tjlm                                          1/1
  Running   1             13d
cee-alpha         logs-retriever-kt6mk                                          1/1
  Running   1             13d
cee-alpha         logs-retriever-rpdd8                                          1/1
  Running   1             13d
cee-alpha         logs-retriever-xkczp                                          1/1
  Running   1             13d
cee-alpha         node-exporter-69gwg                                           1/1
  Running   0             2d4h
cee-alpha         node-exporter-6ghl9                                           1/1
  Running   0             2d4h
cee-alpha         node-exporter-jxrvd                                           1/1
  Running   0             2d4h
cee-alpha         node-exporter-kxztj                                           1/1
  Running   0             2d4h
cee-alpha         node-exporter-xrv7j                                           1/1
  Running   0             2d4h
cee-alpha         ops-center-cee-alpha-ops-center-54899d9984-469p8               4/4
  Running   0             2d4h
cee-alpha         path-provisioner-6hq6h                                        1/1
  Running   1             13d
cee-alpha         path-provisioner-6m52h                                        1/1
  Running   1             13d
cee-alpha         path-provisioner-f7zr6                                        1/1
  Running   1             13d
cee-alpha         path-provisioner-nnmk9                                        1/1
  Running   1             13d
cee-alpha         path-provisioner-vqblj                                        1/1
  Running   1             13d
cee-alpha         pgpool-7b55b577b9-klh6g                                       1/1
  Running   0             2d4h
cee-alpha         pgpool-7b55b577b9-x6hw7                                       1/1
  Running   0             2d4h
cee-alpha         postgres-0                                                    1/1
  Running   0             7d23h
cee-alpha         postgres-1                                                    1/1
  Running   0             7d21h
cee-alpha         postgres-2                                                    1/1
  Running   0             7d21h
cee-alpha         prometheus-hi-res-0                                           4/4
  Running   0             7d23h
cee-alpha         prometheus-hi-res-1                                           4/4
  Running   0             7d21h
```

```
cee-alpha          prometheus-hi-res-2                                          4/4
  Running   0              7d21h
cee-alpha          prometheus-rules-cf645f668-nlsmf                             1/1
  Running   0              7d21h
cee-alpha          prometheus-scrapeconfigs-synch-78f9dd6dbb-hrk94              1/1
  Running   0              7d5h
cee-alpha          pv-manager-84d59c4d4-h6ldw                                   1/1
  Running   0              7d21h
cee-alpha          pv-provisioner-758f6d6875-rkh9l                              1/1
  Running   0              7d21h
cee-alpha          restart-kubelet-888sq                                        1/1
  Running   1              13d
cee-alpha          restart-kubelet-b49nq                                        1/1
  Running   1              13d
cee-alpha          restart-kubelet-h6xbl                                        1/1
  Running   1              13d
cee-alpha          restart-kubelet-hjhwj                                        1/1
  Running   1              13d
cee-alpha          restart-kubelet-r7p7b                                        1/1
  Running   1              13d
cee-alpha          show-tac-manager-86599f9d75-d65rf                            2/2
  Running   0              7d21h
cee-alpha          snmp-trapper-cee-alpha-687nb                                 1/1
  Running   1              8d
cee-alpha          snmp-trapper-cee-alpha-6tkcv                                 1/1
  Running   1              8d
cee-alpha          snmp-trapper-cee-alpha-m9cf5                                 1/1
  Running   1              8d
cee-alpha          snmp-trapper-default-4s24m                                   1/1
  Running   1              8d
cee-alpha          snmp-trapper-default-c9bqn                                   1/1
  Running   1              8d
cee-alpha          snmp-trapper-default-m7swc                                   1/1
  Running   2              8d
cee-alpha          thanos-query-frontend-hi-res-5d8ff67859-h4msn                1/1
  Running   0              7d23h
cee-alpha          thanos-query-frontend-hi-res-5d8ff67859-rfdb4                1/1
  Running   0              7d21h
cee-alpha          thanos-query-frontend-hi-res-5d8ff67859-rw442                1/1
  Running   0              7d21h
cee-alpha          thanos-query-hi-res-68969b56b9-gdkwc                         2/2
  Running   0              7d21h
cee-alpha          thanos-query-hi-res-68969b56b9-snhft                         2/2
  Running   0              7d23h
cee-alpha          thanos-query-hi-res-68969b56b9-t6zft                         2/2
  Running   0              7d21h
chaos-global       chaos-controller-cd9c8fdff-68p58                             2/2
  Running   0              7d18h
chaos-global       chaos-worker-4vzv2                                           2/2
  Running   2              13d
chaos-global       chaos-worker-g28pv                                           2/2
  Running   2              13d
chaos-global       chaos-worker-td2pq                                           2/2
  Running   2              13d
chaos-global       chaos-worker-wg496                                           2/2
  Running   2              13d
chaos-global       chaos-worker-z64jb                                           2/2
  Running   2              13d
chaos-global       ops-center-chaos-global-ops-center-58f8cb786f-2vskv          4/4
  Running   0              2d4h
istio-system       istiod-759c4dd94c-fglr8                                      1/1
  Running   0              7d21h
istio-system       istiod-759c4dd94c-lbkgg                                      1/1
  Running   0              7d21h
```

Sample cnAAA Configuration

```
istio-system      istiod-759c4dd94c-tqztk                                  1/1
  Running   0             7d21h
kube-system       cilium-2rnf4                                             1/1
  Running   1             12d
kube-system       cilium-5hrwc                                             1/1
  Running   1             12d
kube-system       cilium-dqfrw                                             1/1
  Running   1             12d
kube-system       cilium-envoy-5xs9q                                       1/1
  Running   1             13d
kube-system       cilium-envoy-jpcqq                                       1/1
  Running   1             13d
kube-system       cilium-envoy-pqfh4                                       1/1
  Running   1             13d
kube-system       cilium-envoy-w29l9                                       1/1
  Running   1             13d
kube-system       cilium-envoy-wrrsp                                       1/1
  Running   1             13d
kube-system       cilium-nvq2x                                             1/1
  Running   1             12d
kube-system       cilium-operator-6cd9578487-s26j2                         1/1
  Running   2             13d
kube-system       cilium-pkq54                                             1/1
  Running   2             12d
kube-system       cluster-cert-maintainer-ddb8777fb-625qw                  1/1
  Running   0             2d4h
kube-system       coredns-54b8d594bc-bd9xj                                 1/1
  Running   0             2d4h
kube-system       coredns-54b8d594bc-n8zjd                                 1/1
  Running   0             2d4h
kube-system       cpu-sync-js4jr                                           1/1
  Running   0             2d4h
kube-system       cpu-sync-l4mq2                                           1/1
  Running   0             2d4h
kube-system       cpu-sync-zw6rn                                           1/1
  Running   0             2d4h
kube-system       etcd-alpha-master-1                                      1/1
  Running   1             13d
kube-system       etcd-alpha-master-2                                      1/1
  Running   1             13d
kube-system       etcd-alpha-master-3                                      1/1
  Running   1             13d
kube-system       hubble-relay-b5cb8c7f8-kmpz6                             1/1
  Running   0             7d18h
kube-system       hubble-ui-67bb5d95f8-qhzwx                               2/2
  Running   0             7d18h
kube-system       journald-adapter-ck9xn                                   1/1
  Running   1             13d
kube-system       journald-adapter-fm7tb                                   1/1
  Running   1             13d
kube-system       journald-adapter-m6jzn                                   1/1
  Running   1             13d
kube-system       journald-adapter-nf5jf                                   1/1
  Running   1             13d
kube-system       journald-adapter-thhhs                                   1/1
  Running   2             13d
kube-system       kube-apiserver-alpha-master-1                            1/1
  Running   1             13d
kube-system       kube-apiserver-alpha-master-2                            1/1
  Running   1             13d
kube-system       kube-apiserver-alpha-master-3                            1/1
  Running   1             13d
kube-system       kube-controller-manager-alpha-master-1                   1/1
  Running   3             13d
```

```
kube-system       kube-controller-manager-alpha-master-2                          1/1
  Running   1             13d
kube-system       kube-controller-manager-alpha-master-3                          1/1
  Running   1             13d
kube-system       kube-scheduler-alpha-master-1                                   1/1
  Running   2             13d
kube-system       kube-scheduler-alpha-master-2                                   1/1
  Running   1             13d
kube-system       kube-scheduler-alpha-master-3                                   1/1
  Running   1             13d
kube-system       maintainer-gk7dq                                                1/1
  Running   0             2d4h
kube-system       maintainer-ksnzs                                                1/1
  Running   0             2d4h
kube-system       maintainer-rfw49                                                1/1
  Running   0             2d4h
kube-system       maintainer-tqzt5                                                1/1
  Running   0             2d4h
kube-system       maintainer-v86sb                                                1/1
  Running   0             2d4h
kube-system       user-password-monitor-bv96h                                     1/1
  Running   0             2d4h
kube-system       user-password-monitor-dhzn5                                     1/1
  Running   0             2d4h
kube-system       user-password-monitor-mf4hs                                     1/1
  Running   0             2d4h
kube-system       user-password-monitor-pb4tr                                     1/1
  Running   0             2d4h
kube-system       user-password-monitor-pg4pd                                     1/1
  Running   0             2d4h
lbs-alpha         lbs-agent-6dq7r                                                 1/1
  Running   0             7d21h
lbs-alpha         lbs-agent-7lzpv                                                 1/1
  Running   0             7d21h
lbs-alpha         lbs-agent-pw444                                                 1/1
  Running   2             7d21h
lbs-alpha         lbs-agent-qvhml                                                 1/1
  Running   2             7d21h
lbs-alpha         lbs-agent-s9db7                                                 1/1
  Running   2             7d21h
lbs-alpha         lbs-operator-5f6d4b8648-lncld                                   2/2
  Running   0             7d21h
lbs-alpha         ops-center-lbs-alpha-ops-center-57dc5df798-x49zc                4/4
  Running   0             7d5h

nginx-gateway     cndp-gateway-nginx-857d577799-2fpqm            1/1     Running   0
        7d23h
nginx-gateway     cndp-gateway-nginx-857d577799-5p6f4           1/1     Running   0
       7d21h
nginx-gateway     nginx-gateway-fabric-6cdb48b5d8-jqh5v           1/1     Running   0
         7d21h
nginx-gateway     nginx-gateway-fabric-6cdb48b5d8-l25q5       1/1     Running   0
     7d21h
<namespace>         admin-db-0                                                      1/1
    Running   0             7d2h
<namespace>         admin-db-1                                                      1/1
    Running   0             7d2h
<namespace>         base-entitlement-pcf-79797d9b7c-4qgmd                           1/1
    Running   0             7d21h
<namespace>         br-controller-scdb-6d8bf99679-nbnjw                             1/1
    Running   0             7d2h
<namespace>         cdl-ep-session-c1-d0-8ffd988f7-2bg7p                            1/1
    Running   0             7d2h
<namespace>         cdl-ep-session-c1-d0-8ffd988f7-vsbn8                            1/1
```

Sample cnAAA Configuration

```
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m1-0                                        1/1
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m1-1                                        1/1
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m2-0                                        1/1
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m2-1                                        1/1
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m3-0                                        1/1
    Running   0             7d2h
<namespace>          cdl-index-session-c1-m3-1                                        1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m1-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m1-1                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m2-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m2-1                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m3-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m3-1                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m4-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m4-1                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m5-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m5-1                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m6-0                                         1/1
    Running   0             7d2h
<namespace>          cdl-slot-session-c1-m6-1                                         1/1
    Running   0             7d2h
<namespace>          consolidated-aaa-logging-0                                       1/1
    Running   0             7d2h
<namespace>          consolidated-logging-0                                           1/1
    Running   0             7d2h
<namespace>          controlcenter-<namespace>-pcf-engine-app-production-rjio-5bcpjv6b   3/3
     Running   5             7d2h
<namespace>          crd-api-<namespace>-pcf-engine-app-production-rjio-6bcdf5b547b8xt   2/2
     Running   5             7d2h
<namespace>          db-admin-0                                                       1/1
    Running   0             7d2h
<namespace>          db-admin-1                                                       1/1
    Running   0             7d2h
<namespace>          db-admin-2                                                       1/1
    Running   0             7d2h
<namespace>          db-admin-config-0                                                1/1
    Running   0             7d2h
<namespace>          db-admin-config-1                                                1/1
    Running   0             7d2h
<namespace>          db-admin-config-2                                                1/1
    Running   0             7d2h
<namespace>          db-scdb-sdb-subscriber1-0                                        1/1
    Running   0             7d2h
<namespace>          db-scdb-sdb-subscriber1-1                                        1/1
    Running   0             7d2h
<namespace>          db-scdb-sdb-subscriber1-2                                        1/1
    Running   0             7d2h
<namespace>          db-spr-config-0                                                  1/1
```

```
    Running   0             7d2h
<namespace>         db-spr-config-1                                                  1/1
    Running   0             7d2h
<namespace>         db-spr-config-2                                                  1/1
    Running   0             7d2h
<namespace>         db-spr1-0                                                        1/1
    Running   0             7d2h
<namespace>         db-spr1-1                                                        1/1
    Running   0             7d2h
<namespace>         db-spr1-2                                                        1/1
    Running   0             7d2h
<namespace>         etcd-<namespace>-etcd-cluster-0                                   2/2
     Running   0             7d2h
<namespace>         etcd-<namespace>-etcd-cluster-1                                   2/2
     Running   0             7d2h
<namespace>         etcd-<namespace>-etcd-cluster-2                                   2/2
     Running   0             7d2h
<namespace>         grafana-dashboard-cdl-<namespace>-7f7bb99f47-2zr82                1/1
     Running   0             7d2h
<namespace>         grafana-dashboard-etcd-<namespace>-7cfdf8ddc5-wftlc               1/1
     Running   0             7d2h
<namespace>         grafana-dashboard-pcf-585fb9c5b7-h2jd9                           1/1
    Running   0             7d2h
<namespace>         kafka-0                                                          2/2
    Running   0             7d2h
<namespace>         kafka-1                                                          2/2
    Running   0             7d2h
<namespace>         kafka-2                                                          2/2
    Running   0             7d2h
<namespace>         lbvip02-86dcc76fb7-26bb5                                         1/1
    Running   0             7d2h
<namespace>         network-query-222vs                                              1/1
    Running   0             7d2h
<namespace>         network-query-7thw5                                              1/1
    Running   0             7d2h
<namespace>         network-query-9s8cs                                              1/1
    Running   0             7d2h
<namespace>         network-query-csq7f                                              1/1
    Running   0             7d2h
<namespace>         network-query-lqjvp                                              1/1
    Running   0             7d2h
<namespace>         ops-center-<namespace>-ops-center-5b747dc456-8vwkn                4/4
     Running   0             7d5h
<namespace>         patch-server-<namespace>-cnat-cps-infrastructure-57698b85b6-z476p   1/1
     Running   0             7d2h
<namespace>         pcf-engine-<namespace>-pcf-engine-app-production-rjio-8647c6b228l   5/5
     Running   5             7d2h
<namespace>         pcf-engine-<namespace>-pcf-engine-app-production-rjio-8647c6czfzl   5/5
     Running   5             7d2h
<namespace>         pcf-engine-<namespace>-pcf-engine-app-production-rjio-8647c6f9dj7   5/5
     Running   5             7d2h
<namespace>         pcf-prometheus-rules-74fb759cd4-lv7ns                            2/2
    Running   0             7d2h
<namespace>         pcf-utilities-0                                                  1/1
    Running   0             7d2h
<namespace>         policy-builder-<namespace>-pcf-engine-app-production-rjio-588nbvs   1/1
     Running   2             7d2h
<namespace>         prometheus-rules-cdl-7655b49bf-652dk                             1/1
    Running   0             7d2h
<namespace>         prometheus-rules-etcd-7b5c7dc847-kjndn                           1/1
    Running   0             7d2h
<namespace>         radius-ep-0                                                      2/2
    Running   4             7d2h
<namespace>         radius-ep-1                                                      2/2
```

Sample cnAAA Configuration

```
    Running   5             7d2h
<namespace>          radius-ep-2                                                  2/2
    Running   5             7d2h
<namespace>          redis-keystore-0                                             2/2
    Running   0             7d2h
<namespace>          redis-keystore-1                                             2/2
    Running   0             7d2h
<namespace>          redis-queue-0                                                2/2
    Running   0             7d2h
<namespace>          redis-queue-1                                                2/2
    Running   0             7d2h
<namespace>          redis-queue-2                                                2/2
    Running   0             7d2h
<namespace>          rs-controller-admin-b64548888-855lb                          2/2
    Running   0             7d2h
<namespace>          rs-controller-admin-config-66f6b6cfdb-2gxcl                  1/1
    Running   0             7d2h
<namespace>          rs-controller-scdb-sdb-subscriber1-667df76585-n4n7s          1/1
    Running   0             7d2h
<namespace>          rs-controller-spr-config-68f7488794-6v4vc                    1/1
    Running   0             7d2h
<namespace>          rs-controller-spr1-69bbf65dcd-ppsck                          1/1
    Running   0             7d2h
<namespace>          smart-agent-<namespace>-ops-center-78cb748f88-snlkm          1/1
    Running   0             7d21h
<namespace>          svn-0                                                        2/2
    Running   0             7d2h
<namespace>          traceid-0                                                    1/1
    Running   0             7d2h
<namespace>          zookeeper-0                                                  1/1
    Running   0             7d2h
<namespace>          zookeeper-1                                                  1/1
    Running   0             7d2h
<namespace>          zookeeper-2                                                  1/1
    Running   0             7d2h
registry          charts-cee-2025-01-1-i14-0                                   1/1
  Running   0             2d4h
registry          charts-cee-2025-01-1-i14-1                                   1/1
  Running   0             2d4h
registry          charts-cee-2025-01-1-i14-2                                   1/1
  Running   0             2d4h
registry          charts-cpc-2025-01-0-i84-0                                   1/1
  Running   0             7d21h
registry          charts-cpc-2025-01-0-i84-1                                   1/1
  Running   0             7d21h
registry          charts-cpc-2025-01-0-i84-2                                   1/1
  Running   0             7d23h
registry          charts-ulb-2025-01-0-i6-0                                    1/1
  Running   0             7d23h
registry          charts-ulb-2025-01-0-i6-1                                    1/1
  Running   0             7d21h
registry          charts-ulb-2025-01-0-i6-2                                    1/1
  Running   0             7d21h
registry          registry-cee-2025-01-1-i14-0                                 1/1
  Running   0             2d4h
registry          registry-cee-2025-01-1-i14-1                                 1/1
  Running   0             2d4h
registry          registry-cee-2025-01-1-i14-2                                 1/1
  Running   0             2d4h
registry          registry-cpc-2025-01-0-i84-0                                 1/1
  Running   0             7d21h
registry          registry-cpc-2025-01-0-i84-1                                 1/1
  Running   0             7d23h
registry          registry-cpc-2025-01-0-i84-2                                 1/1
```

```
  Running   0              7d21h
registry          registry-ulb-2025-01-0-i6-0                                    1/1
  Running   0              7d23h
registry          registry-ulb-2025-01-0-i6-1                                    1/1
  Running   0              7d21h
registry          registry-ulb-2025-01-0-i6-2                                    1/1
  Running   0              7d21h
registry          software-unpacker-0                                            1/1
  Running   0              7d23h
registry          software-unpacker-1                                            1/1
  Running   0              7d21h
registry          software-unpacker-2                                            1/1
  Running   0              7d21h
smi-certs         ss-cert-provisioner-65f8ffdf96-f2kzq                           1/1
  Running   0              7d21h
smi-ops-control   opscenter-controller-974777585-6l66g                           1/1
  Running   0              7d5h
smi-vips          keepalived-bvjc4                                               3/3
  Running   5              13d
smi-vips          keepalived-ccbjn                                               3/3
  Running   5              13d
smi-vips          keepalived-hkmnj                                               3/3
  Running   5              13d
smi-vips          keepalived-w4bmw                                               3/3
  Running   5              13d
smi-vips          keepalived-x57b2                                               3/3
  Running   5              13d
cloud-user@alpha-master-1:~$


cloud-user@alpha-master-1:~$ kubectl get pods -A | grep 0/
cloud-user@alpha-master-1:~$

cloud-user@alpha-master-1:~$ kubectl get pods -A | awk 'NR==1 || (split($3,a,"/") &&
a[1]!=a[2])'
NAMESPACE         NAME                                                           READY
  STATUS    RESTARTS        AGE
cloud-user@alpha-master-1:~$

cloud-user@alpha-master-1:~$ kubectl get pods -A | grep -v Running
NAMESPACE         NAME                                                           READY
  STATUS    RESTARTS       AGE
cloud-user@alpha-master-1:~$


cloud-user@alpha-master-1:~$ helm ls -n <namespace>
NAME                                            NAMESPACE       REVISION        UPDATED
                                STATUS          CHART
                                   APP VERSION
<namespace>-cnat-cps-infrastructure             <namespace>      1              2025-01-22
 09:25:12.302092085 +0000 UTC deployed
cnat-cps-infrastructure-0.6.10-dev-cpc-2025-01-0050-250113145752-ccfb651
BUILD_2025.01.0.i84
<namespace>-cps-radius-ep                       <namespace>      1              2025-01-22
 09:25:12.314792736 +0000 UTC deployed
cps-radius-ep-0.6.43-dev-cpc-2025-01-0117-250113145837-79379ea
BUILD_2025.01.0.i84
<namespace>-etcd-cluster                        <namespace>      1              2025-01-22
 09:25:12.306024478 +0000 UTC deployed       etcd-cluster-1.6.0-1-6-0154-241127115736-ca72f04
                                BUILD_2025.01.0.i84
<namespace>-network-query                       <namespace>      1              2025-01-22
 09:25:12.31344704 +0000 UTC  deployed
network-query-0.5.4-dev-cpc-2025-01-0085-250113113356-5fecffc
BUILD_2025.01.0.i84
```

```
<namespace>-ngn-datastore                        <namespace>       1              2025-01-22
 09:25:12.312562639 +0000 UTC deployed
ngn-datastore-1.12.0-1-12-1049-250108155638-6f9b7b0
BUILD_2025.01.0.i84
<namespace>-ops-center                           <namespace>       3              2025-01-22
 05:50:50.022064334 +0000 UTC deployed
pcf-ops-center-0.6.32-dev-cpc-2025-01-0540-250114095843-adb26ce
BUILD_2025.01.0.i84
<namespace>-pcf-config                           <namespace>       1              2025-01-22
 09:25:12.314287792 +0000 UTC deployed
pcf-config-0.6.3-dev-cpc-2025-01-0028-250113150230-f686b68
BUILD_2025.01.0.i84
<namespace>-pcf-dashboard                        <namespace>       1              2025-01-22
 09:25:12.305991736 +0000 UTC deployed
pcf-dashboard-0.2.17-dev-cpc-2025-01-0178-250113150137-30d1bc0
BUILD_2025.01.0.i84
<namespace>-pcf-engine-app-production-rjio       <namespace>       1              2025-01-22
 09:25:12.312781104 +0000 UTC deployed
pcf-engine-app-0.9.1-dev-cpc-2025-01-0741-250120133559-dded5d5
BUILD_2025.01.0.i84
<namespace>-pcf-oam-app                          <namespace>       1              2025-01-22
 09:25:12.306231511 +0000 UTC deployed
pcf-oam-app-0.6.2-dev-cpc-2025-01-0021-250114095926-8be6401
BUILD_2025.01.0.i84
<namespace>-pcf-services                         <namespace>       1              2025-01-22
 09:25:12.316334935 +0000 UTC deployed
pcf-services-0.6.17-dev-cpc-2025-01-0081-250113150000-3a1c3c8
BUILD_2025.01.0.i84
cloud-user@alpha-master-1:~$
```

```
cloud-user@alpha-master-1:~$ kubectl get pods -n cpc-alpha | grep db-admin
db-admin-0                                                        1/1     Running   0
     7d2h
db-admin-1                                                        1/1     Running   0
     7d2h
db-admin-2                                                        1/1     Running   0
     7d2h
db-admin-config-0                                                 1/1     Running   0
     7d2h
db-admin-config-1                                                 1/1     Running   0
     7d2h
db-admin-config-2                                                 1/1     Running   0
     7d2h
cloud-user@alpha-master-1:~$
```

Note The Engine pod retains the `'pcf-engine'` designation for backward compatibility.

cnAAA GR configurations

cnAAA Ops-Center Configuration for GR Site01:

This is only a sample configuration of cnAAA Ops-Center settings for GR Site01 provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment. Use plain text password as applicable.

```
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 600

datastore external-endpoints datastore

connection-settings keep-alive keep-alive-time-ms 200

connection-settings channel count 4

connection-settings timeout-ms 500

exit

db global-settings db-replica 3

db global-settings volume-storage-class local

db global-settings backup-settings scp-server host 192.0.2.1

db global-settings backup-settings scp-server port 22

db global-settings backup-settings scp-server user-name cloud-user

db global-settings backup-settings scp-server password
$8$50BpcgMJlN/L3zOKLFOImY/Fs5u1kUT40FYBV8gvKIs=

db global-settings backup-settings scp-server remote-backup-path /home/cloud-user/Backup_user

db scdb replica-name admin-db

port      65005

interface vlan2400

resource cpu limit 3000

resource memory limit 20000

replica-set-label key smi.cisco.com/node-type-5

replica-set-label value admin-db

member-configuration member sdb-rs4-s1-arbiter1

  host    192.0.2.2

  arbiter true

  site    local

exit

member-configuration member sdb-rs4-s1-arbiter2

  host    192.0.2.3

  arbiter true

  site    remote

exit
```

```
member-configuration member sdb-rs4-s1-m1
  host     192.0.2.4
  arbiter  false
  priority 104
  site     local
exit
member-configuration member sdb-rs4-s1-m2
  host     192.0.2.5
  arbiter  false
  priority 103
  site     local
exit
member-configuration member sdb-rs4-s2-m1
  host     192.0.2.6
  arbiter  false
  priority 102
  site     remote
exit
member-configuration member sdb-rs4-s2-m2
  host     192.0.2.7
  arbiter  false
  priority 101
  site     remote
exit
member-configuration member sdb-rs4-s3-arbiter3
  host    192.0.2.8
  arbiter true
  site    remote
exit
exit
db scdb replica-name sdb-spr01
port      65001
```

```
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs1-s1-arbiter1
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs1-s1-arbiter2
  host    192.0.2.3
  arbiter true
  site    remote
exit
member-configuration member sdb-rs1-s1-m1
  host     192.0.2.4
  arbiter  false
  priority 104
  site     local
exit
member-configuration member sdb-rs1-s1-m2
  host     192.0.2.5
  arbiter  false
  priority 103
  site     local
exit
member-configuration member sdb-rs1-s2-m1
  host     192.0.2.6
  arbiter  false
  priority 102
  site     remote
```

```
exit
member-configuration member sdb-rs1-s2-m2
  host     192.0.2.7
  arbiter  false
  priority 101
  site     remote
exit
member-configuration member sdb-rs1-s3-arbiter3
  host    192.0.2.8
  arbiter true
  site    remote
exit
exit
db scdb replica-name sdb-spr02
port      65002
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs2-s1-arbiter1
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs2-s1-arbiter2
  host    192.0.2.3
  arbiter true
  site    remote
exit
member-configuration member sdb-rs2-s1-m1
  host     192.0.2.4
```

```
  arbiter  false
  priority 104
  site     local
exit
member-configuration member sdb-rs2-s1-m2
  host     192.0.2.5
  arbiter  false
  priority 103
  site     local
exit
member-configuration member sdb-rs2-s2-m1
  host     192.0.2.6
  arbiter  false
  priority 102
  site     remote
exit
member-configuration member sdb-rs2-s2-m2
  host     192.0.2.7
  arbiter  false
  priority 101
  site     remote
exit
member-configuration member sdb-rs2-s3-arbiter3
  host    192.0.2.8
  arbiter true
  site    remote
exit
exit
db scdb replica-name sdb-spr03
port      65003
interface vlan2400
resource cpu limit 3000
```

```
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs3-s1-arbiter1
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs3-s1-arbiter2
  host    192.0.2.3
  arbiter true
  site    remote
exit
member-configuration member sdb-rs3-s1-m1
  host     192.0.2.4
  arbiter  false
  priority 104
  site     local
exit
member-configuration member sdb-rs3-s1-m2
  host     192.0.2.5
  arbiter  false
  priority 103
  site     local
exit
member-configuration member sdb-rs3-s2-m1
  host     192.0.2.6
  arbiter  false
  priority 102
  site     remote
exit
member-configuration member sdb-rs3-s2-m2
```

```
  host     192.0.2.7
  arbiter  false
  priority 101
  site     remote
exit
member-configuration member sdb-rs3-s3-arbiter3
  host    192.0.2.8
  arbiter true
  site    remote
exit
exit
debug tracing type DISABLED
debug backup-config backup-type all
debug backup-config username admin
debug backup-config password $8$i3KyeobVckgOGTJUafPaBlHJrAiz+clYTAWqUdUwf1U=
debug backup-config svn-url http://svn/repos/RJIL_PB_APR25
debug backup-config scp-server-user-name root
debug backup-config scp-server-user-ip 192.0.2.9
debug backup-config scp-server-dest-backup-path /opt/Auto-bkp
debug backup-config scp-server-user-password $8$OWABKOFKlplELtLz4yCltDfdHK0Yk+ES/MxP1YK9PSw=
debug backup-config pb-ingress
pb.CPC-gamma-cncps-CPC-engine-app-CPC01production.192.0.2.10.nip.io
debug backup-config crd-ingress
crd-api.CPC-gamma-cncps-CPC-engine-app-CPC01production.192.0.2.10.nip.io
debug logging default-level error
debug logging logger com.broadhop
level error
exit
debug logging logger com.broadhop.balance.impl.autowire.AutowireBalanceManagerBlueprint
level error
exit
debug logging logger com.broadhop.balance.impl.policyintel.PolicyStateStub
level error
```

```
exit
debug logging logger com.broadhop.custrefdata.impl.CustomerReferenceDataManager
level debug
exit
debug logging logger com.broadhop.custrefdata.impl.dao.GenericDao
level debug
exit
debug logging logger com.broadhop.licensing.impl.LicenseCountRequestHandler
level debug
exit
debug logging logger com.broadhop.licensing.impl.LicenseManagerProxy
level debug
exit
debug logging logger com.broadhop.licensing.impl.PrometheusMetricsApi
level debug
exit
debug logging logger com.broadhop.licensing.impl.SmartLicenceMongo
level debug
exit
debug logging logger com.broadhop.radius
level error
exit
debug logging logger com.broadhop.radius.impl.devicemanager
level error
exit
debug logging logger com.broadhop.radius.messages.impl
level debug
exit
debug logging logger com.broadhop.radius.policy.RadiusDelayedStartupManager
level error
exit
debug logging logger com.broadhop.radius.policy.event
```

```
level debug
exit
debug logging logger com.broadhop.radius.policy.event.RadiusMessageDealer
level error
exit
debug logging logger com.broadhop.referencedata.impl
level debug
exit
debug logging logger com.broadhop.referencedata.impl.ReferenceDataManager
level error
exit
debug logging logger com.broadhop.resource.impl
level debug
exit
debug logging logger com.broadhop.resource.impl.ResourceChangeMonitor
level error
exit
debug logging logger com.broadhop.runtime.impl
level error
exit
debug logging logger com.cisco.radius.endpoint.impl
level error
exit
debug logging logger com.cisco.radius.endpoint.utility.LeaderSelector
level debug
exit
debug logging logger com.cisco.radius.smart.licensing.LicenseKPIManager
level debug
exit
debug logging logger policy.engine
level debug
exit
```

```
features patching ingress-enabled true
testing enforce-affinity-rules true
pods-management disable-pods [ redis network-query traceid ]
advance-tuning redis-password $8$GzehhS3rBaAM2cW0bihnzHbSDGL03E/N1L0fekMsHw8=
advance-tuning slice-access-control disabled
advance-tuning slice-based-nf-selection chf disabled
advance-tuning app-resource-control rest-ep memory max-heap-size 4
advance-tuning app-resource-control rest-ep memory new-gen-size 3
advance-tuning async-threading default-worker-threads 25
advance-tuning async-threading default-queue-size 120
advance-tuning async-threading default-processing-threads 12
advance-tuning async-threading http2-connect-timeout-ms 120
advance-tuning async-threading http2-idle-connection-timeout-sec 60
radius bind-ipv4 [ 198.51.100.1 ]
radius bind-ipv6 [ 2001:db8::10:1:42:142 ]
radius replicas 3
radius lbs-service true
radius settings request-timeout-ms 5000
radius settings max-tries  1
radius settings min-processing-time-millis 3000
radius settings backoff-time-millis 1000
radius advance-tuning throttling-limit 100
radius advance-tuning coa-throttlling-for-asr9k-pep true
radius message-authenticator access-accept false
radius message-authenticator accounting-response false
radius device-group ASR9K
default-shared-secret     $8$dqP6Y3ne6pOEg0jjPshAAvOf94SIe1gPlyR/rxdl9kc=
default-coa-shared-secret $8$eAVvx1N1zDAJUVBSsxEwK9ReGIdk16AThB1hfQnNTKg=
coa-port                  3799
coa-retries               3
coa-timeout-seconds       3
device EX-BNG1
```

```
  ip                198.51.100.2
  shared-secret     $8$Pn/NW5P02PfeWgff9E9sqc8dECtU2TSLJ+TnzVdRYOc=
  coa-shared-secret  $8$DC5x/I2KI8UD8VsjXJJHNmWzeDPsaBl2HrXESADjDj0=
  loopback-addresses [ 192.0.2.11 ]
exit
device EX-BNG2
  ip                198.51.100.3
  shared-secret     $8$R5LNawhpHvCtjHdemcGdDkVHv4/Gebh28uhm2ylZYjU=
  coa-shared-secret  $8$vQhG90kIXVTwmLaqLhj2rxZ3XyxDGW/nKf0ZeIw7ifU=
  loopback-addresses [ 192.0.2.11 ]
exit
device EX-BNG3
  ip                198.51.100.4
  shared-secret     $8$qDifuNqPGwsCsGODUT+NA6Ypf7G51izfB449i5WIXyk=
  coa-shared-secret  $8$ZVnedWz3Xq+h7QsxlzuFXaIPBY3QoWpFUezjC8EsO4A=
  loopback-addresses [ 192.0.2.11 ]
exit
exit
radius server-group grp1
servers DEL_OCS
  primary             198.51.100.5
  secondary           198.51.100.6
  nas-ip              192.0.2.12
  accounting-port     1805
  authorization-port  1802
  auth-protocol       PAP
  radius-password     $8$rgcQTgLEudJdUtfCgLwAYC16ktAi4Xu7qjF/riX02Dg=
  shared-secret       $8$30UyK5lB2EwfzEPYWrXS/sn+DWfOXzm6XV3s7D8Wq+0=
  timeout-seconds     3
  test-message        false
  test-userid         test
  test-password       $8$RpDHAuN/zCZYroXNasrvul/ocQHve4GVGxiLnFVTLg0=
```

```
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers GUJ_OCS
  primary              198.51.100.7
  secondary            198.51.100.8
  nas-ip               192.0.2.12
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$uABWJQgerqpneO6gd2AGaSAIxCitNpH69PG+9jFW9fg=
  shared-secret        $8$OU/UwXacpI9i/N9ftnCAbqa04CA0MjDX2LnnQW2eLEM=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$Ua4NmzgUbJ3EA1v86eE2fMMtF3pm1Ghb0WueZOepXhA=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers LUCK_OCS
  primary              198.51.100.9
  secondary            198.51.100.10
  nas-ip               192.0.2.12
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$U4EOrGRDyN8nq96aVVsZlei3DkGbdEfqQQL444t0ufw=
  shared-secret        $8$Tgq1MKmmRs195IeZNEWreCQpOIeWiJhN62sF5Cp8W84=
```

```
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$9F9a+1KCLxXLIg0mpeSPasKBNDpuBvYTQIFOgYNGy5c=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers PBHR_OCS
  primary              198.51.100.11
  secondary            198.51.100.12
  nas-ip               192.0.2.12
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$cA7xG56sPrhWYYccYGHCuozy7t8CAtc/q5TCTX0CZLo=
  shared-secret        $8$37deIR+G0mdMhH9rSn3OfQ6scAtHYvtQgWF5o19zBak=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$Ko8EAV1Sdv/6OgMwJHGAiiGeJeRTt/MI+jQocQS3Dvs=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers PassiveMZ-12997
  primary              198.51.100.13
  secondary            198.51.100.14
  nas-ip               192.0.2.12
  accounting-port      1805
```

```
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$IqebiUpWoztzVa+L0dVdXxBEGApCZ9Q68Jj3T8HKHWM=
  shared-secret        $8$5XrbqUxj+vdcZCyfVEglaw6NHMCD87D1RyMUxjEqJfM=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$W192Tn+h2sf7DW8dc4dy/O+kpCNX5S2/SjerSIMmFp4=
  thread-pool-size     300
  server-type          offline
  max-proxy-queue-size 50000
  retries              3
exit
exit
radius async-threading-configuration default-processing-threads 100
radius async-threading-configuration default-action-priority 5
radius async-threading-configuration default-action-threads 100
radius async-threading-configuration default-action-queue-size 400000
radius async-threading-configuration default-action-drop-oldest-when-full true
radius properties PROMETHEUS_PORT
value 9099
exit
radius properties backOffRetryCoA.CoANackErrorCause
value 405,506,1001
exit
radius properties backOffRetryCoA.constantdelayInSeconds
value 60
exit
radius properties backOffRetryCoA.maxRetransmission
value 3
exit
radius properties com.broadhop.pep.ipv6.enable.feature
```

```
value true
exit
radius properties enable.radius.auth
value true
exit
radius properties grpc.executors
value 40
exit
radius properties grpc.request.proxyAccounting.timeoutMillies
value 3000
exit
radius properties grpc.timeoutMs.processing
value 15000
exit
radius properties io.netty.eventLoopThreads
value 100
exit
radius properties parallelChannelCount
value 10
exit
radius properties prometheusPort
value 9099
exit
radius properties radiusCorePoolSize
value 40
exit
radius properties radiusMaxQueue
value 4000
exit
radius properties smart.license.kpi.interval.mins
value 1
exit
```

```
radius properties traps.tps
value 5000
exit
radius properties udpMaxQueue
value 4000
exit
radius properties udpPoolSize
value 40
exit
api unified engine-group CPC01production
api unified externalIPs [ 198.51.100.15 ]
api unified external-port 8080
engine CPC01production
config-lock           false
replicas              3
subversion-run-url    http://svn/repos/run
subversion-config-url http://svn/repos/configuration
tracing-service-name  CPC-engine
crdapi crd-mongo-cache-refresh-interval 10000
crdapi admin-db primary 192.0.2.4
crdapi admin-db secondary 192.0.2.5
crdapi admin-db port 65005
properties GeoSiteName
  value gamma_site1
exit
properties balanceKeyHashAvpName
  value SprKeyHash
exit
properties broadcast.tps
  value 500
exit
properties cc.ua.soap.url
```

```
  value http://127.0.0.1:8080/apirouter
exit
properties coaThrottlingPerBng
  value true
exit
properties com.broadhop.SrgBngSwitchOverEnable
  value true
exit
properties com.broadhop.cc.login.details.feature
  value true
exit
properties com.broadhop.domain.ipv6.enable.feature
  value true
exit
properties com.broadhop.pb.login.details.feature
  value true
exit
properties com.broadhop.pb.publish.audit.feature
  value true
exit
properties com.broadhop.pep.ipv6.enable.feature
  value
exit
license smart reservation
system mode running
```

cnAAA Ops Center Configuration for GR Site02:

This is only a sample configuration of cnAAA Ops-Center settings for Site02 provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

```
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 600
datastore external-endpoints datastore
connection-settings keep-alive keep-alive-time-ms 200
```

```
connection-settings channel count 4
connection-settings timeout-ms 500
exit
db global-settings db-replica 3
db global-settings volume-storage-class local
db scdb replica-name admin-db
port      65005
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type-5
replica-set-label value admin-db
member-configuration member sdb-rs4-s1-arbiter1
  host    192.0.2.1
  arbiter true
  site    remote
exit
member-configuration member sdb-rs4-s1-arbiter2
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs4-s1-m1
  host     192.0.2.3
  arbiter  false
  priority 104
  site     remote
exit
member-configuration member sdb-rs4-s1-m2
  host     192.0.2.4
  arbiter  false
  priority 103
```

```
  site     remote
exit
member-configuration member sdb-rs4-s2-m1
  host     192.0.2.5
  arbiter  false
  priority 102
  site     local
exit
member-configuration member sdb-rs4-s2-m2
  host     192.0.2.6
  arbiter  false
  priority 101
  site     local
exit
member-configuration member sdb-rs4-s3-arbiter3
  host    192.0.2.7
  arbiter true
  site    remote
exit
exit
db scdb replica-name sdb-spr01
port      65001
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs1-s1-arbiter1
  host    192.0.2.1
  arbiter true
  site    remote
exit
```

```
member-configuration member sdb-rs1-s1-arbiter2
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs1-s1-m1
  host     192.0.2.3
  arbiter  false
  priority 104
  site     remote
exit
member-configuration member sdb-rs1-s1-m2
  host     192.0.2.4
  arbiter  false
  priority 103
  site     remote
exit
member-configuration member sdb-rs1-s2-m1
  host     192.0.2.5
  arbiter  false
  priority 102
  site     local
exit
member-configuration member sdb-rs1-s2-m2
  host     192.0.2.6
  arbiter  false
  priority 101
  site     local
exit
member-configuration member sdb-rs1-s3-arbiter3
  host    192.0.2.7
  arbiter true
```

```
  site    remote
exit
exit
db scdb replica-name sdb-spr02
port      65002
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs2-s1-arbiter1
  host    192.0.2.1
  arbiter true
  site    remote
exit
member-configuration member sdb-rs2-s1-arbiter2
  host    192.0.2.2
  arbiter true
  site    local
exit
member-configuration member sdb-rs2-s1-m1
  host     192.0.2.3
  arbiter  false
  priority 104
  site     remote
exit
member-configuration member sdb-rs2-s1-m2
  host     192.0.2.4
  arbiter  false
  priority 103
  site     remote
exit
```

```
member-configuration member sdb-rs2-s2-m1
  host     192.0.2.5
  arbiter  false
  priority 102
  site     local
exit
member-configuration member sdb-rs2-s2-m2
  host     192.0.2.6
  arbiter  false
  priority 101
  site     local
exit
member-configuration member sdb-rs2-s3-arbiter3
  host    192.0.2.7
  arbiter true
  site    remote
exit
exit
db scdb replica-name sdb-spr03
port      65003
interface vlan2400
resource cpu limit 3000
resource memory limit 20000
replica-set-label key smi.cisco.com/node-type
replica-set-label value oam
member-configuration member sdb-rs3-s1-arbiter1
  host    192.0.2.1
  arbiter true
  site    remote
exit
member-configuration member sdb-rs3-s1-arbiter2
  host    192.0.2.2
```

```
  arbiter true
  site    local
exit
member-configuration member sdb-rs3-s1-m1
  host     192.0.2.3
  arbiter  false
  priority 104
  site     remote
exit
member-configuration member sdb-rs3-s1-m2
  host     192.0.2.4
  arbiter  false
  priority 103
  site     remote
exit
member-configuration member sdb-rs3-s2-m1
  host     192.0.2.5
  arbiter  false
  priority 102
  site     local
exit
member-configuration member sdb-rs3-s2-m2
  host     192.0.2.6
  arbiter  false
  priority 101
  site     local
exit
member-configuration member sdb-rs3-s3-arbiter3
  host    192.0.2.7
  arbiter true
  site    remote
exit
```

```
exit
debug tracing type DISABLED
debug logging default-level error
debug logging logger com.broadhop
level error
exit
debug logging logger com.broadhop.licensing.impl.LicenseCountRequestHandler
level debug
exit
debug logging logger com.broadhop.licensing.impl.LicenseManagerProxy
level debug
exit
debug logging logger com.broadhop.licensing.impl.PrometheusMetricsApi
level error
exit
debug logging logger com.broadhop.licensing.impl.SmartLicenceMongo
level debug
exit
debug logging logger com.broadhop.policy.impl
level debug
exit
debug logging logger com.broadhop.radius
level error
exit
debug logging logger com.broadhop.radius.impl.devicemanager
level debug
exit
debug logging logger com.broadhop.radius.messages.impl
level error
exit
debug logging logger com.broadhop.radius.policy.RadiusDelayedStartupManager
level error
```

```
exit
debug logging logger com.broadhop.radius.policy.event
level error
exit
debug logging logger com.broadhop.radius.policy.event.RadiusMessageDealer
level error
exit
debug logging logger com.cisco.radius.actions.impl
level error
exit
debug logging logger com.cisco.radius.endpoint.impl
level error
exit
debug logging logger com.cisco.radius.endpoint.impl.RadiusEndpointMessageListener
level debug
exit
debug logging logger com.cisco.radius.endpoint.impl.util
level error
exit
debug logging logger com.cisco.radius.endpoint.utility.LeaderSelector
level debug
exit
debug logging logger com.cisco.radius.smart.licensing.LicenseKPIManager
level debug
exit
debug logging logger policy.engine
level debug
exit
features patching ingress-enabled true
testing enforce-affinity-rules true
pods-management disable-pods [ redis network-query traceid ]
advance-tuning redis-password $8$jEoUSZxGgkcGfFCR95Z5ZRE1Bdp7J+eP0lr4VDPbC2Y=
```

```
advance-tuning slice-access-control disabled
advance-tuning async-threading default-worker-threads 25
advance-tuning async-threading default-queue-size 120
advance-tuning async-threading default-processing-threads 12
advance-tuning async-threading http2-connect-timeout-ms 120
advance-tuning async-threading http2-idle-connection-timeout-sec 60
radius bind-ipv4 [ 192.0.2.8 ]
radius bind-ipv6 [ 2001:db8::1 ]
radius replicas 1
radius lbs-service true
radius settings request-timeout-ms 5000
radius settings max-tries  1
radius settings min-processing-time-millis 1000
radius settings backoff-time-millis 1000
radius advance-tuning throttling-limit 10
radius advance-tuning coa-throttlling-for-asr9k-pep true
radius message-authenticator access-accept false
radius message-authenticator accounting-response false
radius device-group ASR9K
default-shared-secret     $8$Hgpy1vW0YuzKbgt9iWFAPj+IqcFLFxgbkp9s+TMJIgM=
default-coa-shared-secret $8$83Hhbxao8HfFjIYoKn8z8kBXs44TPcPgIFhRiB9BXNc=
coa-port                  3799
coa-retries               3
coa-timeout-seconds       3
device EX-BNG1
  ip                 192.0.2.9
  shared-secret      $8$F3+sGL0FjR5R+O4XGqo+J0o3s4E2RpfbQgzRQtCF9zY=
  coa-shared-secret  $8$uOAYc3Vt1lKdrlp0EHaym2B+FpH+b984OFJiRLPQSL4=
  loopback-addresses [ 198.51.100.1 ]
exit
device EX-BNG2
  ip                 192.0.2.10
```

```
  shared-secret      $8$HFvy2NknPjuGYoHqXzleHZaCTed76osXr7nbJEPLb4o=
  coa-shared-secret  $8$cIfI0dHd2i9RLgTfgJUrnfIKUgmwkWa5Vj6vbEG0yfE=
  loopback-addresses [ 198.51.100.1 ]
exit
device EX-BNG3
  ip                 192.0.2.11
  shared-secret      $8$q1n/El8lGiMhofFVvbQJw0zMy0BMhAe4uSFVUAZdp9M=
  coa-shared-secret  $8$w9IRXdD2zuGjWnAKRSLgf8RFxqcywcz0pcUUu7/TVeM=
  loopback-addresses [ 198.51.100.1 ]
exit
exit
radius server-group grp1
servers DEL_OCS
  primary              192.0.2.12
  secondary            192.0.2.13
  nas-ip               192.0.2.14
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$inbnxcbFpx90KAbRlb0CPmiyfY4Fm3+ur6WbHqs4aGo=
  shared-secret        $8$d14/JWy9migM6wnVoV1Pm/fK6aiBxX3Lrm5DRfz3liU=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$+KEGV6qWQBITNvrI/nrjsm/gyEJnPxbIXdMvur9IapA=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers GUJ_OCS
  primary              192.0.2.15
```

```
  secondary            192.0.2.13
  nas-ip               192.0.2.14
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$AhwQuyuBStb2KM+RkRAV2OQJJQ37wF8GW2KihZuAsck=
  shared-secret        $8$cmiPSOzIXdcKjLPMQFJffifa8h/39GwDTfmt50hXMhk=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$A6k9SnXu5rMnLGqfnUB4fKmGWITYg9zk7fzqgFuLgJk=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers LUCK_OCS
  primary              192.0.2.16
  secondary            192.0.2.13
  nas-ip               192.0.2.14
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$6gYFdcOroQsOiD8wKmgZh37+RdZEyp+oTzw00PKimG8=
  shared-secret        $8$+pLtFGo2a7OGmRyFaIRWIokQ1ILoAQwcMXyUTxHorA8=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$EHftY2XyKpc3j0WRddtlFAbt28pEQ/jhhi6PthXVgTc=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
```

```
  retries              3
exit
servers PBHR_OCS
  primary              192.0.2.17
  secondary            192.0.2.13
  nas-ip               192.0.2.14
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$QuSmQ22UIA5yxnQQG1xCfZOzH7ufESZXx6Y+E6LERiA=
  shared-secret        $8$kVWgx4fd1MTlzoA3EnuD5+L7w9ELN1CIqHNEHW/qgzI=
  timeout-seconds      3
  test-message         false
  test-userid          test
  test-password        $8$TTCtNmQLRhuc5gBUrtkx/vvpkFqwK+JxhhxlcIiPOY4=
  thread-pool-size     300
  server-type          online
  max-proxy-queue-size 50000
  retries              3
exit
servers PassiveMZ-12997
  primary              192.0.2.18
  secondary            192.0.2.13
  nas-ip               192.0.2.19
  accounting-port      1805
  authorization-port   1802
  auth-protocol        PAP
  radius-password      $8$T4jkp4FNUES0nu1SylJI8+6ERUktlwF/2n6WM2V3r9s=
  shared-secret        $8$zvaQd++DsvjdLbqhRGhglZc4SFCTmLUIHmr0mI5Dl0k=
  timeout-seconds      3
  test-message         false
  test-userid          test
```

```
  test-password        $8$LBlPEuGt7crFTasVX86yB63TIwlAV+oLq2D0yDauMaI=
  thread-pool-size     300
  server-type          offline
  max-proxy-queue-size 50000
  retries              3
exit
exit
radius async-threading-configuration default-processing-threads 100
radius async-threading-configuration default-action-priority 5
radius async-threading-configuration default-action-threads 100
radius async-threading-configuration default-action-queue-size 400000
radius async-threading-configuration default-action-drop-oldest-when-full true
radius properties PROMETHEUS_PORT
value 9099
exit
radius properties backOffRetryCoA.CoANackErrorCause
value 405,506,1001
exit
radius properties backOffRetryCoA.constantdelayInSeconds
value 60
exit
radius properties backOffRetryCoA.maxRetransmission
value 3
exit
radius properties com.broadhop.pep.ipv6.enable.feature
value true
exit
radius properties enable.radius.auth
value true
exit
radius properties grpc.executors
value 5
```

```
exit
radius properties grpc.request.proxyAccounting.timeoutMillies
value 3000
exit
radius properties grpc.timeoutMs.processing
value 15000
exit
radius properties io.netty.eventLoopThreads
value 100
exit
radius properties parallelChannelCount
value 10
exit
radius properties prometheusPort
value 9099
exit
radius properties radiusCorePoolSize
value 20
exit
radius properties radiusMaxQueue
value 4000
exit
radius properties smart.license.kpi.interval.mins
value 1
exit
radius properties traps.tps
value 5000
exit
radius properties udpMaxQueue
value 4000
exit
radius properties udpPoolSize
```

```
value 40
exit
api unified engine-group CPC02production
api unified externalIPs [ 192.0.2.20 ]
api unified external-port 8080
engine CPC02production
config-lock           false
replicas              1
subversion-run-url    http://svn/repos/run
subversion-config-url http://svn/repos/configuration
tracing-service-name  CPC-engine
crdapi crd-mongo-cache-refresh-interval 10000
crdapi admin-db primary 192.0.2.3
crdapi admin-db secondary 192.0.2.4
crdapi admin-db port 65005
properties GeoSiteName
  value delta_site2
exit
properties balanceKeyHashAvpName
  value SprKeyHash
exit
properties broadcast.tps
  value 500
exit
properties cc.ua.soap.url
  value http://127.0.0.1:8080/apirouter
exit
properties coaThrottlingPerBng
  value true
exit
properties com.broadhop.SrgBngSwitchOverEnable
  value true
```

```
exit
properties com.broadhop.domain.ipv6.enable.feature
  value true
exit
properties com.broadhop.pb.login.details.feature
  value true
exit
properties com.broadhop.pb.publish.audit.feature
  value true
exit
properties com.broadhop.pep.ipv6.enable.feature
  value true
exit
properties com.broadhop.servicemismatchenable
  value false
exit
properties com.cisco.engine.log.type
  value 2
exit
properties conflict.resolve.attempts
  value 2
exit
properties conflict.tps
  value 5
exit
properties cpc.smart.license.enabled
  value true
exit
properties crd.mongo.cache.refresh.interval
  value 10000
exit
properties dbSocketTimeout
```

```
  value 1000
exit
properties enable.conflict.merge
  value true
exit
properties enable.radius.aaa.key.building
  value true
exit
properties grpc.executors
  value 5
exit
properties grpc.request.BackoffTimeMillis
  value 100
exit
properties grpc.request.MinProcessingTimeMillis
  value 11
exit
properties grpc.request.asyncCoA.timeoutMillies
  value 15000
exit
properties grpc.request.bundledCoA.timeoutMillies
  value 15000
exit
properties grpc.request.getMaxTries
  value 1
exit
properties grpc.request.proxyAccounting.timeoutMillies
  value 5000
exit
properties grpc.request.timeoutMillies
  value 10
exit
```

```
properties inboundMessageQueueSize
  value 4000
exit
properties inboundMessageSlaMs
  value 3000
exit
properties maxHash
  value 3
exit
properties messageSlaMs
  value 3000
exit
properties mongo.connections.per.host.default
  value 300
exit
properties mongo.threads.allowed.to.wait.for.connection
  value 1000
exit
properties CPC.actions.sync.timeoutMs.default
  value 10000
exit
properties queryEachSiteForSearchSubscribers
  value true
exit
properties radius.engine.service.port.timeout
  value 5000
exit
properties radius.warmup.duration.sec
  value 30
exit
properties radius.warmup.event.dealy.ms
  value 20
```

```
exit
properties radius.warmup.request.per.thread
  value 50
exit
properties radius.warmup.threads
  value 5
exit
properties radius.warmup.unified.soap.url
  value http://localhost:8080/ua/soap
exit
properties replaceFullNameInSearchSubscribers
  value true
exit
properties returnBalance
  value false
exit
properties sessioncount.query.interval.mins
  value 1
exit
properties skip.service.evaluation.on.delete
  value false
exit
properties skipped.device.mgrs
  value RX_5G_TGPP,N7_TGPP,N5_TGPP,N
exit
license smart reservation
system mode running
```

Sample ULB Configuration

ULB Ops Center Configuration for GR Site01

This is only a sample configuration of ULB settings for GR Site01 provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment. Use plain text password as applicable.

```
logging level error

logging name lbs-agent.egress-mgr.app level error

logging name lbs-agent.egress-mgr.calico level error

logging name lbs-agent.egress-mgr.egressmgr level error

logging name lbs-agent.egress-mgr.ipt-reconciler level error

logging name lbs-agent.egress-mgr.iptables level error

logging name lbs-agent.egress-mgr.iptables-oper level error

logging name lbs-agent.egress-mgr.lbs-iptables level error

logging name lbs-agent.egress-mgr.listener level error

logging name lbs_operator.app.app level error

logging name lbs_operator.app.empcrd level error

logging name lbs_operator.app.lbcrd level error

logging name lbs_operator.app.service level error

cilium bpf auth-map-max    4194303

cilium bpf fragments-map-max 65536

cilium bpf lb-affinity-map-max 4194304

cilium bpf lb-map-max      4194304

cilium bpf lb-rev-nat-map-max 4194304

cilium bpf lb-service-backend-map-max 4194303

cilium bpf lb-service-map-max 4194304

cilium bpf lb-source-range-map-max 4194304

cilium bpf neigh-global-max 4194304

cilium bpf node-map-max    4194303

cilium bpf policy-map-max  16383

system mode running
```

ULB Ops Center Configuration for GR Site02

This is only a sample configuration of ULB settings for GR Site02 provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment. Use plain text password as applicable.

```
logging level error

logging name lbs-agent.egress-mgr.app level error
```

```
logging name lbs-agent.egress-mgr.calico level error
logging name lbs-agent.egress-mgr.egressmgr level error
logging name lbs-agent.egress-mgr.ipt-reconciler level error
logging name lbs-agent.egress-mgr.iptables level error
logging name lbs-agent.egress-mgr.iptables-oper level error
logging name lbs-agent.egress-mgr.lbs-iptables level error
logging name lbs-agent.egress-mgr.listener level error
logging name lbs_operator.app.app level error
logging name lbs_operator.app.empcrd level error
logging name lbs_operator.app.lbcrd level error
logging name lbs_operator.app.service level error
cilium bpf auth-map-max    4194303
cilium bpf fragments-map-max 65536
cilium bpf lb-affinity-map-max 4194304
cilium bpf lb-map-max      4194304
cilium bpf lb-rev-nat-map-max 4194304
cilium bpf lb-service-backend-map-max 4194303
cilium bpf lb-service-map-max 4194304
cilium bpf lb-source-range-map-max 4194304
cilium bpf neigh-global-max 4194304
cilium bpf node-map-max    4194303
cilium bpf policy-map-max  16383
system mode running
```