CPS vDRA Operations Guide, Release 18.2.0 (Restricted Release) (1)

First Published: 2018-05-11

Last Modified: 2018-07-30

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.

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: www.cisco.com go trademarks. 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)

© 2018 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

About this guide

This document describes how to manage CPS vDRA using graphical interfaces, APIs, CLI commands, etc.

Audience

This guide is best used by these readers:

- Network administrators
- Network engineers
- Network operators
- System administrators

This document assumes a general understanding of network architecture, configuration, and operations.

Additional Support

For further documentation and support:

- Contact your Cisco Systems, Inc. technical representative.
- Call the Cisco Systems, Inc. technical support number.
- Write to Cisco Systems, Inc. at support@cisco.com.
- Refer to support matrix at https://www.cisco.com/c/en/us/support/index.html and to other documents related to Cisco Policy Suite.

Conventions (all documentation)

This document uses the following conventions.

Conventions	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
courier font	Terminal sessions and information the system displays appear in courier font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning IMPORTANT SAFETY INSTRUCTIONS.

Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Note Regulatory: Provided for additional information and to comply with regulatory and customer requirements.

Obtaining Documentation and Submitting a Service Request

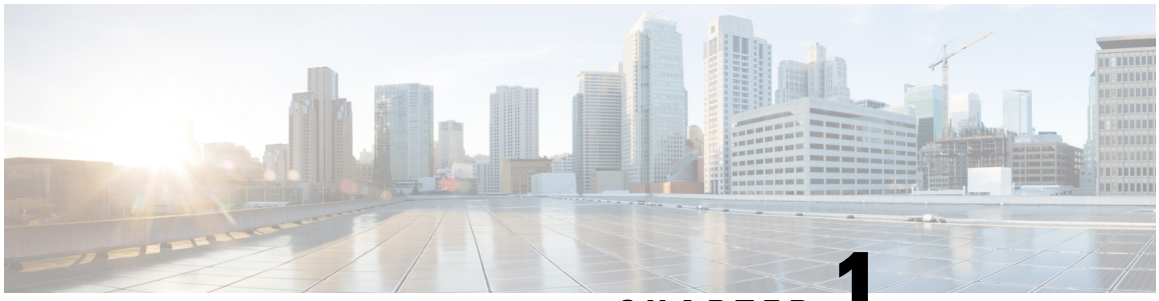
For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the . RSS feeds are a free service.

RESTRICTED RELEASE

Important This is a Short Term Support (STS) release with availability and use restrictions. Contact your Cisco Account or Support representatives for more information.

RESTRICTED RELEASE



CHAPTER 1

Managing CPS vDRA Cluster

Accessing CPS vDRA Management CLI

There are two options for accessing the CPS vDRA Management CLI.

Access Via Web Browser

Perform the following steps to access the CPS vDRA Management CLI:

Step 1 Enter the following URL in Firefox or Chrome:

https://<masterip>/

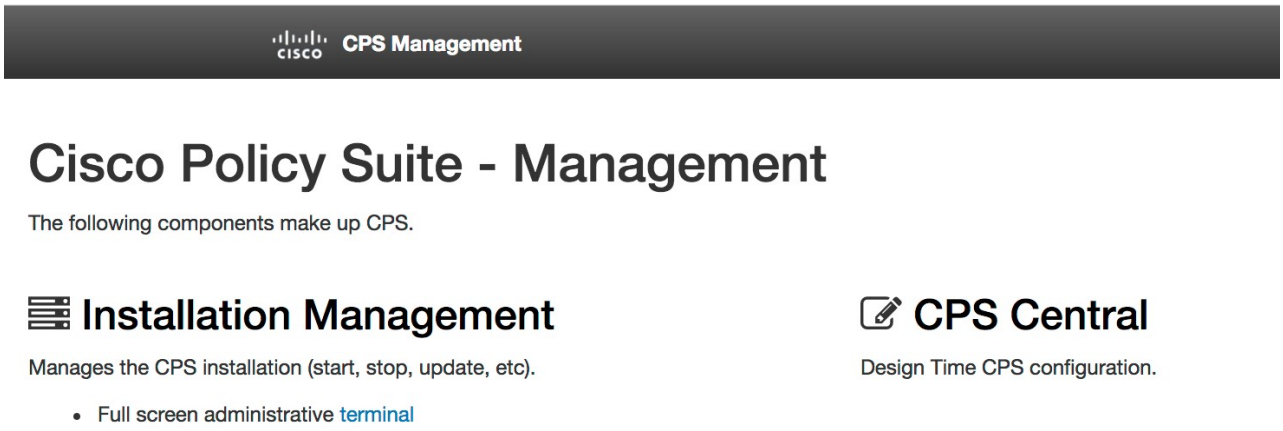
Step 2 Login to the application using your user ID and password.

Step 3 Follow the Installation Management hyperlink in the following screen:

Figure 1: CPS DRA Login

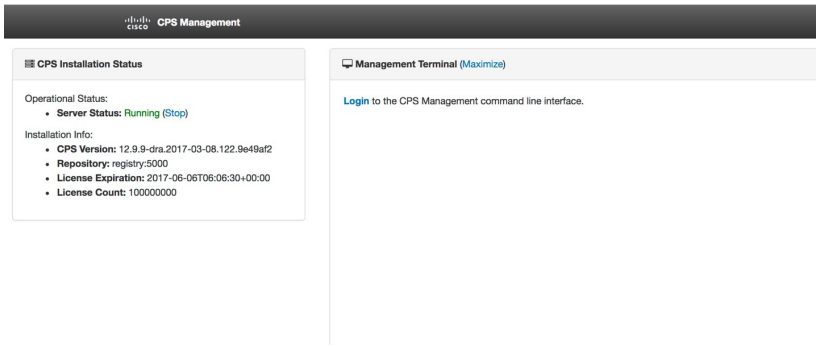
Step 4 In the Management screen, click the **Login** link to display the in-browser terminal window.

Figure 2: Installation Management



Step 5 Login with a valid user name and password.

Figure 3: Management Terminal Link



Access Via SSH

Access is available to the CPS vDRA via SSH listening on port 2024 of the master virtual machine. This port must be open in the OpenStack security rules in order to access the Management CLI via SSH.

Starting CPS vDRA Cluster

A CPS vDRA cluster is a self-organizing cluster that does not require operator actions to configure the system when you follow the instructions found in the installation guide. The system self-organizes by following the algorithm:

1. The cluster master node is started and bootstraps the Docker engine, an embedded Docker registry, the Weave overlay network, and the CPS vDRA scheduling application.
2. The worker nodes are started either after the master node is started or in parallel. The bootstrapping of the Docker engine and Weave overlay network point back to the master node.
3. The scheduling function on the master node begins an auto discovery function on engine startup of the Docker engines that have joined the Weave overlay network.
4. For each engine discovered, the system queries the Docker engine configuration to discover the node identifier and the role within the cluster that the engine will perform. The roles are used by the scheduling function to map application services to the appropriate virtual machines.
    1. The CPS vDRA application (for both Policy DRA and IMS DRA solutions) supports the following roles:
        1. master – This is always the master scheduling node.
        2. control-a[b] – This is a control node that works in concert with the other control node and the master node to provide OAM support for the application.
        3. diameter-endpoint – This is the node where all diameter traffic terminals.
        4. binding-worker – This is the node where binding/slf queries are executed.
    2. The vDRA Binding and SLF application supports the following roles:

1. master – This is always the master scheduling node.
2. control-a[b] – control node that works in concert with the other control nodes and the master node to provide OAM support for the application.
3. persistence-router – node where binding/slf queries are routed.
4. persistence-db – nodes where the binding database replica sets are located.

5. As the Docker engines are registered, the scheduling application begins executing a controlled startup by starting modules as the underlying engines become available.
   1. A module is a set of interrelated services that are started, stopped and scaled as a set of related processes. These processes are either collocated on the same virtual machine or across multiple virtual machines. There are three type of modules that exist:
      1. infrastructure – These are core modules that are not shutdown when the application shuts down.
      2. application – These are modules that are removed when the application is shutdown.
      3. External – These are external services that are installed on the system and whose images are built and loaded outside of the system. See the **scheduling external-service** command for more information on configuring external services.

Stopping Application Services In CPS vDRA Cluster

The modules of type “application” can be shut down in a controlled manner by running the **system stop** command. This command will unload all modules in reverse run-level order and stop the associated running Docker services.

Starting Services In CPS vDRA Cluster

The modules of type “application” can be started in a controlled manner by running the **system start** command. This command will start all modules in run-level order and schedule the underlying Docker services on the registered Docker engines.

Stopping External Services In CPS vDRA Cluster

The modules of type “external” can be shut down in a controlled manner by running the **system disable-external-services** command. This command will unload all modules in reverse run-level order and stop the associated running Docker services.

Starting External Services In CPS vDRA Cluster

The modules of type “external” can be shut down in a controlled manner by running the **system enable-external-services** command. This command will unload all modules in reverse run-level order and stop the associated running Docker services.

Restarting An Individual Docker Service

Perform the following steps to restart an individual docker service:

Step 1 Run the **show docker service** command to locate the container ID of the service to restart.

```
scheduler# show docker service

PENALTY
MODULE           INSTANCE  NAME               VERSION          ENGINE  CONTAINER ID       STATE
BOX      MESSAGE
-------------------------------------------------------------------------------------------------------------------
admin-db         1         mongo-admin-a      3.4.0.0          aio     mongo-admin-a      HEALTHY
false    -
admin-db         1         mongo-admin-arb    3.4.0.0          aio     mongo-admin-arb    HEALTHY
false    -
admin-db         1         mongo-admin-b      3.4.0.0          aio     mongo-admin-b      HEALTHY
false    -
admin-db         1         mongo-admin-setup  12.9.9-SNAPSHOT  aio     mongo-admin-setup  HEALTHY
false    -
consul           1         consul-1           12.9.9-SNAPSHOT  aio     consul-1           HEALTHY
false    -
consul           1         consul-2           12.9.9-SNAPSHOT  aio     consul-2           HEALTHY
false    -
consul           1         consul-3           12.9.9-SNAPSHOT  aio     consul-3           HEALTHY
false    -
foobar           1         foobar             3.2.6.0          aio     foobar             HEALTHY
false    -
grafana          1         grafana            12.9.9-SNAPSHOT  aio     grafana            HEALTHY
false    -
haproxy-common   1         haproxy-common     12.9.9-SNAPSHOT  aio     haproxy-common-s1  HEALTHY
false    -
orchestrator-ui  1         orchestrator-ui    12.9.9-SNAPSHOT  aio     orchestrator-ui    HEALTHY
false    -
subversion       1         svn                12.9.9-SNAPSHOT  aio     svn                HEALTHY
false    -
```

Step 2 Using the provided container-id, run the **docker restart container-id** *container-id* command. This will issue a non-graceful stop on the Docker container and move the state of the container to ABORTED. The container will stay in this state for 10 seconds before restarting.

Step 3 Verify the health of the restarted docker service by running the **show docker service** command again and waiting for the service to progress into the HEALTHY state. Optionally the log of the individual container can be followed by running the **monitor log container** *container-id* using the same container id from Step 2.

CPS External Authentication and Authorization

CPS system supports LDAP external authentication and authorization.

Based on Conf-D group configurations, CPS roles are assigned to the applications running on CPS cluster.

The following command configures the gid mapping for various roles.

```
admin@orchestrator(config)# external-aaa pam gid-mapping
 1000 policy-admin
admin@orchestrator(config-gid-mapping-1000/policy-admin)# commit
Commit complete
```

You can also view the status of configuration with the following command:

```
admin@orchestrator# show running-config external-aaa | tab
```

Sample Output:

```
admin@orchestrator# show running-config external-aaa | tab
GID GROUP
--------------------
1000 policy-admin
```

Conf-D Group to CPS Roles Description

The following table describes the CPS roles and Conf-D groups of applications/services:

Table 1: Conf-D Group to CPS Roles Description

Application/Service	CPS Role	Conf-D Groups
Control center	SUMADMIN	crd-read-write
Control center	READONLY	crd-read-only
Policy Builder	READ&WRITE	policy-admin
Policy Builder	READ	*
SVN	READ&WRITE	policy-admin
SVN	READ	*
Grafana	Admin	grafana-admin
Grafana	Editor	grafana-editor
Grafana	Viewer	*

***** Indicates all authenticated users

Bulkstats conf-D group: sftp daemon running on port 2026 retrieves all statistics within the /var/broadhop/stats directory. Users associated to the “bulkstats” or “admin” group are able to retrieve statistics.

Oper conf-D group is not used.

vDRA Containers

The following table describes the modules, containers, and the respective VM location in vDRA:

Module	Container	VM on which container runs	Description
admin-db	mongo-admin-a	master	Stores the collection of system and CRD related configurations
admin-db	mongo-admin-b	control-a	Stores the collection of system and CRD related configurations
admin-db	mongo-admin-c	control-b	Stores the collection of system and CRD related configurations
admin-db	mongo-admin-setup	master	Sets up the mongo database cluster across the master, control-a and control-b
binding	binding	dra-worker	Provides functionality for handling the requests from diameter-endpoint to binding database and vice versa
cc-monitor	cc-monitor	control-a, control-b	Manages haproxy instance for memcached servers and also for the collection of consolidated qns and engine logs.
configuration-engine	configuration-engine	control-a	Maintains confd configuration engine details
consul	consul-1	master	Service discovery and configuration
consul	consul-2	control-a	Service discovery and configuration
consul	consul-3	control-b	Service discovery and configuration

Module	Container	VM on which container runs	Description
control-plane	control-plane	master,control-a, control-b	Passes topology information via control messages from publishers to subscribers.
control-plane	control-plane-monitor	master,control-a, control-b	Monitors server running in control-plane container and restarts if the same is not responsive or down
diameter-endpoint	diameter-endpoint	dra-director	Maintains Diameter endpoint inbound and outbound connections,message handling and routing function.
diameter-endpoint	diameter-redis-q-a	dra-director	Facilitate inter process communication of application messages across nodes.
diameter-endpoint	diameter-redis-q-a-monitor	dra-director	Monitor IPC server process in "diameter-redis-q-a" and restarts if the same is not responsive or down
diameter-endpoint	global-control-plane	dra-director	Passes topology information via control messages from publishers to subscribers across DRA installations
diameter-endpoint	interface-mover	dra-director	Provides functionality for moving of SCTP interface from host to inside container.
diameter-endpoint	socket-forwarder	dra-director	Forwards the socket bind connections from host to inside container
docker-registry	registry	master	Internal docker registry for storing and distributing of images running on the system
docker-registry	registry-extra	master	Utility container to support docker registry

Module	Container	VM on which container runs	Description
grafana	grafana	control-a/control-b	Provides a graphical or text-based representation of statistics and counters collected in the Prometheus database
haproxy-common	haproxy-api	on all nodes except dra-worker	haproxy instance for the load balancing of API servers
haproxy-common	haproxy-common	on all nodes except dra-worker	Common haproxy instance for the load balancing of Policy Builder, Grafana, orchestrator CLI and UI, API, CC, etc.
haproxy-int-api	haproxy-int-api	control-a	haproxy instance for the load balancing of internal API servers.
haproxy-prometheus	haproxy-prometheus	control-a/control-b	haproxy instance for the load balancing of Prometheus services.
memcached-vip	lbvip02		In-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering. Intended for use in speeding up dynamic web applications by alleviating database load.
mongo-node	mongo	master, control-a, control-b	Maintains sharded clusters for managing of huge data.
mongo-node	mongo-monitor	master, control-a, control-b	Monitoring of Mongo shards that run on Mongo containers.
mongo-node	mongo-status	master	Monitoring of Mongo database configurations

Module	Container	VM on which container runs	Description
monitoring	collectd-host	All	The collection utility collectd is used for collecting and storing statistics from each VM to the centralized collection nodes on the control-A and control-B virtual machines. The centralized collector writes the collected data to output CSV files.
monitoring	dnsmasq	All	Used for internal DNS forwarding and caching
monitoring	dnsmasq-monitor	All	Monitoring and managing dnsmasq container
monitoring	docker-host-info	All	Sytem utility container used for executing all system related commands
monitoring	keepalived	All	Manages the VIPs configured via VRRP protocol
monitoring	keepalived-monitor	All	Monitors the keepalived process running on the system and starts the keepalived process with the given VIP name
monitoring	node-exporter	All	Exporter for the System metrics like CPU, RAM, DISK etc
monitoring	node-exporter-monitor	All	Monitoring of node exporter container
monitoring	ntpd	All	NTP service for time synchronization that runs either realtime or on client process based on the reachability of the NTP server .

Module	Container	VM on which container runs	Description
orchestrator	orchestrator	master	1. Creates and maintains docker engines 2. Schedules and manages docker services 3. All system operations like upgrade, downgrades 4. CLI operations 5. Alert and SNMP functionalities et
orchestrator-backup-a	orchestrator-backup-a	control-a	Provides high availability support for the functionalities carried out by the orchestrator.
orchestrator-backup-b	orchestrator-backup-b	control-b	Provides high availability support for the functionalities carried out by the orchestrator.
orchestrator-ui	orchestrator-ui	master, control-a, control-b	To access the management console via HTTP
policy-builder	policy-builder	control-a, control-b	Service configurations and policy rules
prometheus	blackbox-exporter	master, control-a, control-b	**Note**: Will be obselete in future releases, as ICMP statistics are now collected from orchestrator
prometheus	prometheus-hi-res	master, control-a, control-b	Monitors the system at 5-second intervals with 24-hour history
prometheus	prometheus-planning	master, control-a, control-b	Monitors the system at 120-second intervals with 365-day history
prometheus	prometheus-trending	master, control-a, control-b	Monitors the system at 20-second intervals with 30-day history

Module	Container	VM on which container runs	Description
prometheus	statistics-gathering	master, control-a, control-b	Collection of statistics related to java applications as bulk stats
stats	collectd-jmx	control-a, control-b	Collection of statistics related to jmx using collectd
stats	stats-relay	control-a, control-b	Collection of statistics related to relay interfaces using collectd
stats	stats-sftp	control-a, control-b	Collection of statistics related to sftp
subversion	svn	control-a/control-b	Maintains all the CPS policy configurations and has repositories in which files can be created, updated and deleted
zvision	haproxy-zvision	master, control-a, control-b	haproxy instance for the load balancing of zvision servers
zvision	zvision	master, control-a, control-b	Provides functionality of Zing VM monitoring

Installing New Software Images

When a new ISO is provided with software, you need to perform the following steps to upgrade the current system software:

Step 1 Download the ISO image from CCO site.

Step 2 Copy the ISO to DRA VNF /data/iso/staged-isos.

Step 3 Run the following commands:

```
system software iso load category product file <ISO file name>
 activate true
```

```
show system software available-versions
```

Step 4 Repeat the steps for the DRA database ISO.

Upgrading To A New Software Version

Perform the following steps to upgrade to a new software version:

Step 1 Run the following command:

```
system software iso load category product file cisco-policy-dra.iso activate true
```

Step 2 In the Management CLI, run **show system software available-versions** to determine if the correct version of has been uploaded:

```
scheduler# show system software available-versions
VERSION
---------------------------------
12.9.9-dra.2017-03-08.122.9e49af2
```

Step 3 In the Management CLI, run the **system upgrade version** command to upgrade to the version found in Step 2:

```
scheduler# system upgrade version 12.9.9-dra.2017-03-08.122.9e49af2
```

At this point the application will begin downloading the new scheduling and application images from the on-board Docker Registry. The download will take several seconds and the scheduler application will disconnect and restart. You must re-login after the disconnect occurs.

Step 4 In the Management CLI, run the **show scheduling status** command to validate the progress of the upgrade.

Aborting An Upgrade

If an in-progress upgrade needs to be aborted, run the **system abort-upgrade** command. This will immediately stop all scheduling activities. Reverting to the previous versions is triggered by the downgrade to a previous software version procedure.

Downgrading To Previous Software Version

Perform the following steps to downgrade to a previous software version:

Step 1 Select the qualifier for the version you want to downgrade and then activate the ISOs for downgrading as shown in the following example:

```
admin@orchestrator[mpsl14fdrm01v]# system abort-upgrade
admin@orchestrator[mpsl14fdrm01v]# show system software iso details
| tab
CATEGORY NAME           VERSION QUALIFIER                   CREATED                    ACTIVE SIZE
 MB
-------------------------------------------------------------------------------------------------------
product cisco-policy-dra 13.1.1 dra.2017-12-06.1366.b800a6d 2018-03-02T23:37:21.848+00:00 false
1339.99
product cisco-policy-dra 13.1.1 dra.2018-02-28.1793.f618c58 2018-03-12T22:42:19.225+00:00 false
1341.93
product cisco-policy-dra 13.1.1 dra.2018-03-28.1938.f618c58 2018-04-13T21:10:34.872+00:00 true
```

```
1342.13
admin@orchestrator[mpsl14fdrm01v]# system software iso activate category product
name cisco-policy-dra version 13.1.1 qualifier dra.2018-02-28.1793.f618c58
```

Step 2 In the Management CLI, run the **show system software available-versions** to determine if the correct version has been uploaded:

```
scheduler# show system software available-versions
VERSION
---------------------------------
12.9.9-dra.2017-03-08.122.9e49af2
```

Step 3 In the Management CLI, run the **system downgrade version** command to upgrade to the version found in Step 3:

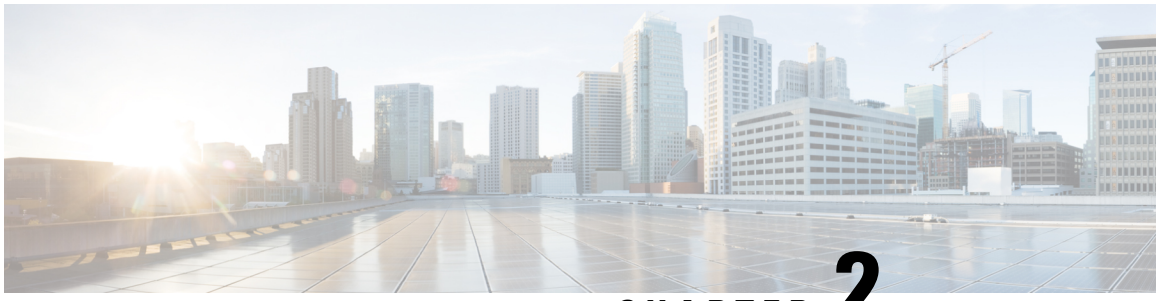
```
scheduler# system downgrade version 12.9.9-dra.2017-03-08.122.9e49af2
```

At this point the application will begin downloading the new scheduling and application images from the on-board Docker Registry. The download will take several seconds and the scheduler application will disconnect and restart. You must re-login after the disconnect occurs.

Step 4 In the Management CLI, run the **show scheduling status** command to validate the progress of the upgrade.

Aborting A Downgrade

If an in-progress downgrade needs to be aborted, run the **system abort-downgrade** command. This will immediately stop all scheduling activities. Reverting to the previous versions is triggered by the upgrading to a new software version procedure.



CHAPTER 2

Prometheus and Grafana

Introduction

CPS system, application statistics and Key Performance Indicators (KPI) are collected by the system and are displayed using a browser-based graphical metrics tool. This chapter provides a high-level overview of the tools CPS uses to collect and display these statistics.

Prometheus

Prometheus is an application that is used to actively gather statistics and trigger alerts from the running virtual machines and application services. The CPS vDRA cluster deploys the following Prometheus services on each control node and on the master node:

- Prometheus Hi-Res – this instance of the Prometheus service is monitoring the system at 5 second intervals with 24-hour history
- Prometheus Trending – this instance of the Prometheus service is monitoring the system at 20 second intervals with 30-day history
- Prometheus Planning – this instance of the Prometheus service is monitoring the system at 120 second intervals with 365-day history

Internally, the Prometheus servers scrape statistics from target statistics sources on a regular basis. The following target data sources are included:

- Host Node Exporter for Host VM statistics.
- Mongo DB Exporter for Database statistics.
- Application Statistics.

In addition to scrapping, statistics in the Prometheus servers can be configured using the Management CLI alert rule command to trigger alerts on error conditions. In this scenario, a user defines the alert rule and the configuration for that rule is pushed into the Prometheus servers. It can generate SNMPv2 and SNMPv3 alarm based on the NMS destination configured in the system. You can configure multiple SNMP destination (SNMPv2, SNMPv3) to receive the alarms at multiple NMS.

Note Currently, SNMP get and walk facility is not supported.

For more information on Prometheus, refer https://prometheus.io/.

Grafana

Grafana is a third-party metrics dashboard and graph editor provided with CPS 7.0 and higher. Grafana provides a graphical or text-based representation of statistics and counters collected in the Prometheus database.

Additional Grafana Documentation

This chapter provides information about the CPS implementation of Grafana. For more information about Grafana, or access the general Grafana documentation, refer to: http://docs.grafana.org.

Data Source Supported

The CPS implementation uses the Prometheus data source and does not use graphite for queries. This requires the definition of queries to use the Prometheus query format as defined in https://prometheus.io/docs/querying/basics/.

Note If the control VM that hosts Grafana goes down, then the Prometheus data also not available during that downtime after the same control VM (hosting Grafana) is back. This results in some missing data. As a workaround, you can add the Prometheus datasource of other control VM in Grafana UI that was up during that downtime and view the missing statistics.

Manage Grafana Users

Note In Grafana, admin users can invite new users by email or a link. However, this is not supported in CPS vDRA.

Perform the following to add a new Grafana:

1. Enter config mode

```
scheduler# config
Entering configuration mode terminal
scheduler(config)#
```

2. Enter the **aaa authentication** command to create the user:

```
scheduler(config)# aaa authentication users user test2 gid 100 uid 9000 homedir / password
 testpassword ssh_keydir /
scheduler(config-user-test2)# commit
scheduler(config-user-test2)# exit
```

Note The **gid**, **uid**, **homedir** and **ssh_keydir** are required but not used by the application.

Add User To A Viewer Operational Group

In config mode, add the user to the “oper” group and commit as follows:

```
scheduler(config)# nacm groups group oper user-name test2
scheduler(config-group-oper)# commit
```

Add User To A Grafana Editor Group

In config mode, add the user to the “grafana-editor” group and commit as follows:

```
scheduler(config)# nacm groups group grafana-editor user-name test2
scheduler(config-group-grafana-editor)# commit
```

Add User To A Grafana Admin Group

In config mode, add the user to the “grafana-admin” group and commit as follows:

```
scheduler(config)# nacm groups group grafana-admin user-name test2
scheduler(config-group-grafana-admin)# commit
```

Change A Grafana Users Password

In the Management CLI, issue the **aaa authentication users** user **change-password** command as follows:

```
scheduler# aaa authentication users user test2 change-password
Value for 'old-password' (<string>): ************
Value for 'new-password' (<string>): ********
Value for 'confirm-password' (<string>): ********
scheduler#
System message at 2017-03-08 21:17:18...
Commit performed by system via system using system.
```

Specify Access Restrictions for a Group

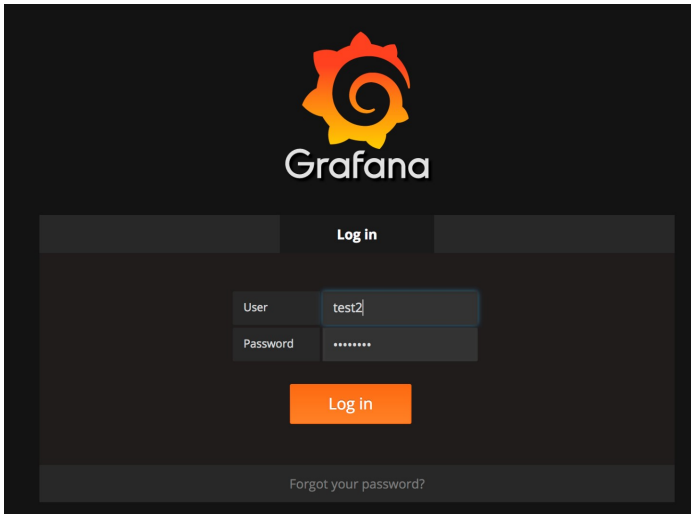
For more information, see the nacm rule-list command.

Connect to Grafana

Use the following URL to access Grafana and enter the user name and password:

https://<masterip>/grafana/

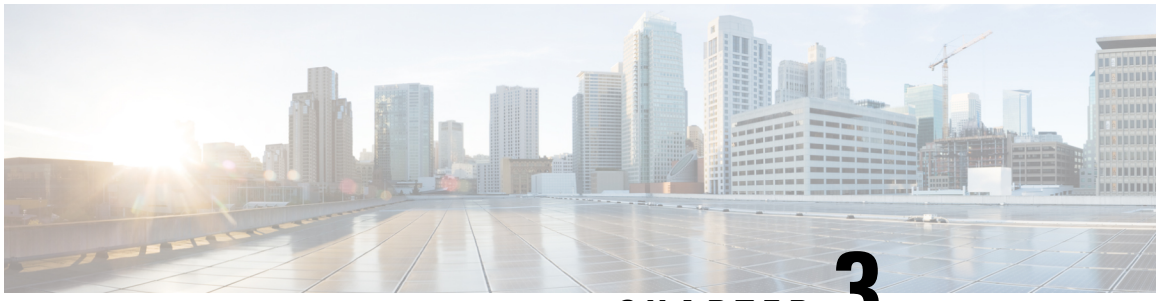
Figure 4: Grafana Login



Grafana Roles

The following types of user roles are supported:

- Admin: An admin user can view, update and create dashboards. Also, the admin can edit and add data sources and organization users.
- Viewer: A viewer can only view dashboards and cannot not save or create them.
- Editor: An editor can view, update and create dashboards.



CHAPTER 3

Managing CPS Interfaces And APIs

CPS vDRA Interfaces And APIs

CPS vDRA includes various application APIs to configure and manage the application.

CRD REST API

Purpose

The Custom Reference Data (CRD) REST API enables the query of creation, deletion, and update of CRD table data without the need to access the Control Center GUI. The CRD APIs are available using an HTTP REST interface.

URL and Port

https:// <*master ip or control node* >:443/custrefdata

Protocol

HTTPS

Accounts and Roles

Security for the CRD REST API is accomplished by using HTTP basic authentication to support read-only and read-write access to the CRD REST API.

Assigning a Read-Only User

Use the **nacm groups group** command to assign the user to the "crd-read-only" group.

For Example, nacm groups group crd-read-only user-name oper

Grafana

Purpose

Grafana is a metrics dashboard and graph editor used to display graphical representations of system, application KPIs, bulkstats of various CPS components.

URL and Port

https:// <*master ip or control node* >:443/grafana

Protocol

HTTPS

Accounts and Roles

For more information on adding or deleting these user accounts, refer to the *Prometheus and Grafana* chapter in this guide.

JMX Interface

Purpose

Java Management Extension (JMX) interface can be used for managing and monitoring applications and system objects.

Resources to be managed or monitored are represented by objects called managed beans (mbeans). MBean represents a resource running in JVM and external applications can interact with mbeans through the use of JMX connectors and protocol adapters for collecting statistics (pull), for getting/setting application configurations (push/pull), and notifying events like faults or state changes(push).

CLI Access

Perform the following steps to access the jmxterm:

1. Run **docker connect** *container-id*.
2. Run the jmxterm command from the CLI prompt to bring up the jmx terminal

Port

All applications run on port 9045.

This port is not exposed externally.

Accounts and Roles

Not applicable.

OSGi Console

Purpose

CPS is based on Open Service Gateway initiative (OSGi) and OSGi console is a command-line shell which can be used for analyzing problems at OSGi layer of the application. It may become necessary to connect to the OSGi console to execute specific commands. These commands are not documented in this guide but the connection process is described below.

CLI Access

Use the following command to access the OSGi console:

1. Run the command **docker connect** *container-id*.
2. `telnet <ip> <port>`

Ports

All applications run on port 9091 within the executing container.

This port is not exposed externally.

Accounts and Roles

Not applicable.

Policy Builder GUI

Purpose

Policy Builder is the alternative web-based client interface for the configuration of the Cisco Policy Suite.

URL and Port

https://<*master or control ip*>/pb

Protocol

HTTPS

Accounts and Roles

Assigning a Read-Only User

It is not necessary to assign a read-only role. Any valid user that can login will have read-only access.

Assigning a Read-Write User

Use the **nacm groups group** command to assign the user to the "policy-admin" group.

For example, nacm groups group policy-admin user-name admin

DRA Central GUI

Purpose

DRA Central is the primary web-based client interface for the configuration and operational control of the CPS vDRA.

URL and Port

https://<*master or control ip*>/central/dra/

Protocol

HTTPS

Accounts and Roles

Assigning a Read-Only User

Use the **nacm groups group** command to assign the user to the "policy-ro" group.

Assigning a Read-Write User

Use the **nacm groups group** command to assign the user to the "policy-admin" group.

For example: `nacm groups group policy-admin user-name admin`

SVN Interface

Apache ™ Subversion (SVN) is the versioning and revision control system used within CPS. It maintains all the CPS policy configurations and has repositories in which files can be created, updated and deleted. SVN maintains the file difference each time any change is made to a file on the server and for each change it generates a revision number.

In general, most interactions with SVN are performed via Policy Builder.

CLI Access

From a remote machine with the SVN client installed, use the following command to access SVN:

Access all files from the server as follows:

```
svn checkout --username <username> --password <password> <SVN Repository URL> <Local Path>
```

Example:

```
svn checkout --username admin --password admin https://<master ip or control ip>/repos/
```

If <*Local Path*>is not provided, files are checked out to the current directory.

Check-in the changed files to the server as follows:

```
svn commit --username <username> --password <password> <Local Path> -m "modified config"
```

Example:

```
svn commit --username broadhop --password broadhop /root/configuration -m "modified config"
```

Update local copy to latest from SVN:

```
svn update <Local Path>
```

Example:

```
svn update /root/configuration/
```

Check current revision of files:

```
svn info <Local Path>
```

Example:

```
svn info /root/configuration/
```

Use **svn --help** for a list of other commands.

Protocol

HTTPS

URL and Port

https://<*master or control ip*>/repos/

Accounts and Roles

Assigning a Read-Only User

It is not necessary to assign a read-only role. Any valid user that can login will have read-only access.

Assigning a Read-Write User

Use the **nacm groups group** command to assign the user to the "policy-admin" group.

For example, nacm groups group policy-admin user-name admin

Multi-user Policy Builder

Multiple users can be logged into Policy Builder at the same time.

In the event that two users attempt to make changes on same screen and one user saves their changes to the client repository, the other user may receive errors. In such cases the user must return to the login page, revert the configuration, and repeat their changes.

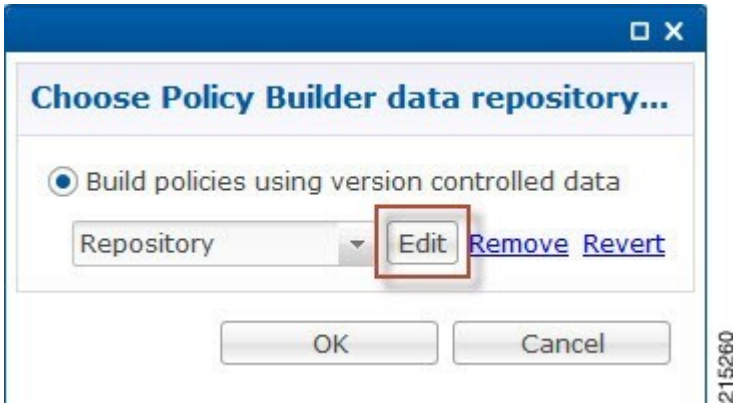
Revert Configuration

You can revert the configuration if changes since the last publish/save to client repository are not wanted.

This can also be necessary in the case of a ' syn conflict ' error where both pcrfclient01 and pcrfclient02 are in use at the same time by different users and publish/save to client repository changes to the same file. The effect of reverting changes is that all changes since the publish/save to client repository will be undone.

Step 1 On the Policy Builder login screen, verify the user for which changes need to be reverted is correct. This can be done by clicking **Edit** and verifying that the Username and Password fields are correct.

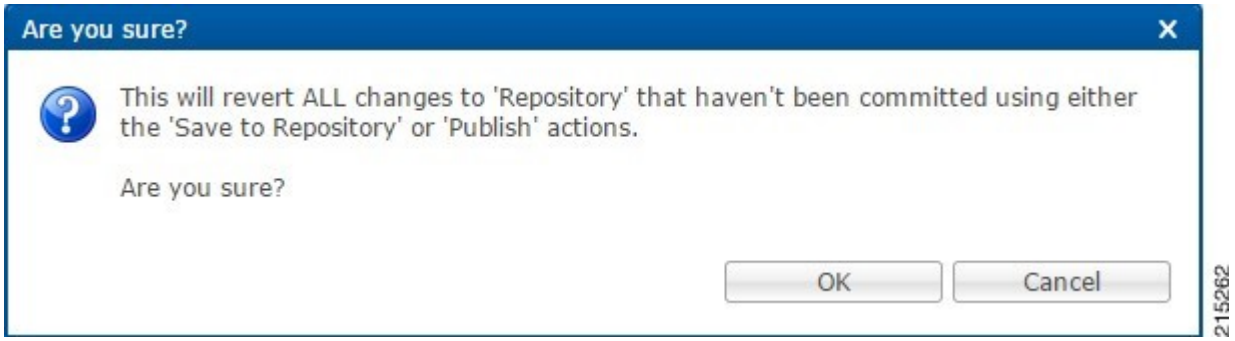
Figure 5: Verifying User



Step 2 Click **Revert**.

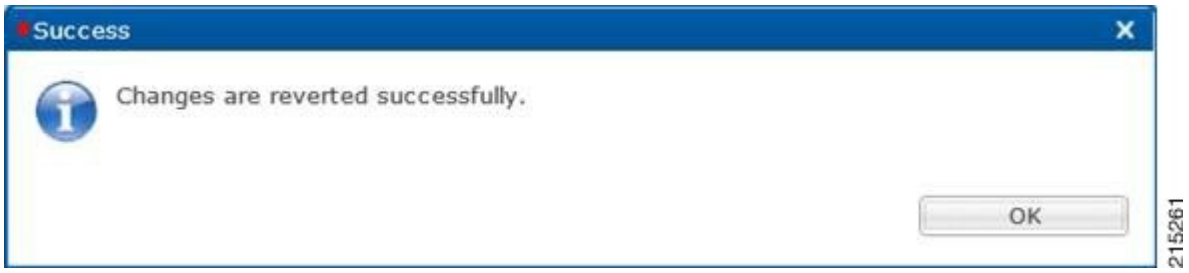
The following confirmation dialog opens.

Figure 6: Revert Confirmation Message



Step 3 Click **OK** to revert back to the earlier configuration. The following dialog confirms that the changes are reverted successfully.

Figure 7: Success Confirmation Message



Publishing Data

This section describes publishing Cisco Policy Builder data to the Cisco Policy Server. Publishing data occurs in the Cisco Policy Builder client interface, but affects the Cisco Policy Server.

Cisco Policy Builder manages data stored in two areas:

- The Client Repository stores data captured from the Policy Builder GUI in Subversion. This is a place where trial configurations can be developed and saved without affecting the operation of the Cisco Policy Builder server data.

  The default URL is http://svn/repos/configuration.

- The Server Repository is where a copy of the client repository is created/updated and where the CPS picks up changes. This is done on Publish from Policy Builder.

  The default URL is http://svn/repos/run.

CRD APIs

You can use Custom Reference Data (CRD) APIs to query, create, delete, and update CRD table data without the need to utilize the Control Center interface. The CRD APIs are available via a REST interface.

Limitations

These APIs allow maintenance of the actual data rows in the table. They do not allow the creation of new tables or the addition of new columns. Table creation and changes to the table structure must be completed via the Policy Builder application.

All table names should be in lowercase alphanumeric to utilize these APIs. Spaces and special characters are not allowed in the table name.

- Table names containing uppercase characters will return code 400 Bad Request.
- Spaces in the name are not allowed and will be flagged as an error in Policy Builder.
- Special characters even when escaped or encoded in ASCII throw errors with the APIs and should not be used.

Setup Requirements

Policy Builder

Step 1 Log in to the Policy Builder.

Step 2 Select **Reference Data** tab.

Step 3 Select **Systems** from the left pane.

Step 4 Select and expand your system name.

Step 5 Select **Plugin Configurations** (or a sub cluster or instance), a Custom Reference Data Configuration plugin configuration is defined.

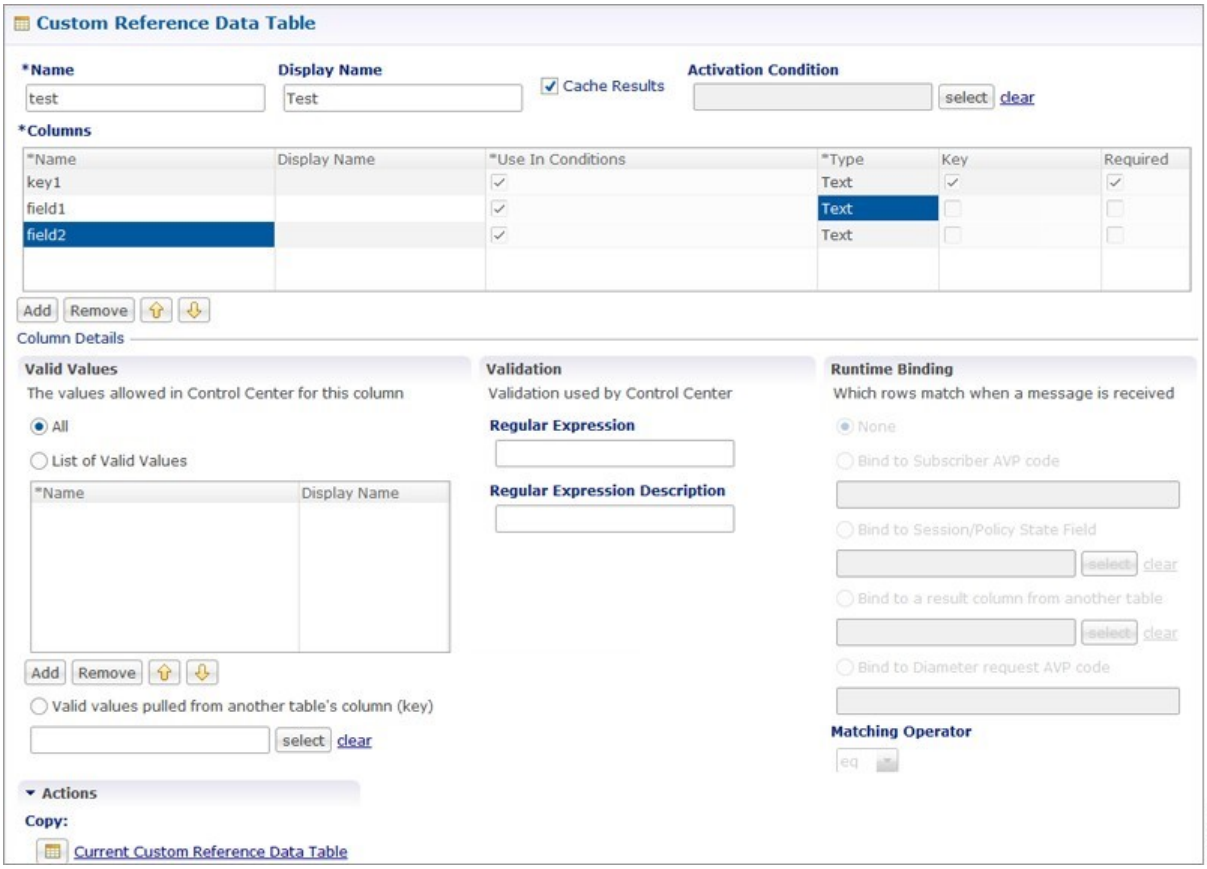
The following parameters can be configured under **Custom Reference Data Configuration**:

Table 2: Custom Reference Data Configuration Parameters

Parameter	Description
Primary Database IP Address	IP address of the primary sessionmgr database. This should remain the default of mongo-admin-a.
Secondary Database IP Address	Optional, this field is the IP address of a secondary, backup, or failover sessionmgr database. This should remain the default of mongo-admin-b.
Database Port	Port number of the sessionmgr. It should be the same for both the primary and secondary databases.
Db Read Preference	Read preference describes how sessionmgr clients route read operations to members of a replica set. You can select from the following drop-down list: • Primary: Default mode. All operations read from the current replica set primary. • PrimaryPreferred: In most situations, operations read from the primary but if it is unavailable, operations read from secondary members. • Secondary: All operations read from the secondary members of the replica set. • SecondaryPreferred: In most situations, operations read from secondary members but if no secondary members are available, operations read from the primary. For more information, refer to http://docs.mongodb.org/manual/core/read-preference/.
Connection Per Host	Number of connections that are allowed per database host. Default value is 100.

Step 6 In **Reference Data** tab > **Custom ReferenceData Tables**, at least one Custom Reference Data Table must be defined.

Figure 8: Custom Reference Data Configuration



The following parameters can be configured under Custom Reference Data Table:

Table 3: Custom Reference Data Table Parameters

Field	Description
Name	Name of the table that will be stored in the database. It should start with alphanumeric characters, should be lowercase or uppercase but not mixed case, and should not start with numbers, no special characters are allowed, use "_" to separate words. For example, logical_apn = GOOD, logicalAPN = BAD, no_spaces.
Display Name	Name of the table that will be displayed in Control Center.
Cache Results	Indicates if the tables should be cached in memory and should be checked for production.
Activation Condition	Custom Reference Data Trigger that needs to be true before evaluating this table. It can be used to create multiple tables with the same data depending on conditions or to improve performance if tables do not need to be evaluated based on initial conditions.
Best Match	When enabled, it allows '*' to be used in the values of the data and the best matching row is returned.

Field	Description
Evaluation Order	Indicates the order the tables within the search table group should be evaluated. Starting with 0 and increasing.
Columns	
Name	Name of the column in the database.
Display Name	More readable display name.
Use In Conditions	Represents the availability of the row for conditions in Policies or Use Case Templates. There is a performance cost to having these enabled, so it is recommended to disable unless they are required.
Type	Determines the values in the control centre as described below: • Text: Value can be any character. For example, example123!. • Number: Value should be a whole number. For example, 1234. • Decimal: Value can be any number. For example, 1.234. • True/False: Value can be true or false. For example, true. • Date: Value should be a date without time component. For example, May 17th 2020. • DateTime: Value should be a date and time. For example, May 17th, 2020 5:00pm.
Key	Indicates that this column is all or part of the key for the table that makes this row unique. By default, a key is required. Keys also are allowed to set the Runtime Binding fields to populate this data from the current message/session. Typically, keys are bound to data from the current session (APN, RAT Type) and other values are derived from them. Keys can also be set to a value derived from another custom reference data table.
Required	Indicates whether this field will be marked required in Control Center. A key is always required.
Column Details	
Valid Values	
All	All the values of the type selected by the user.
List of Valid	A list of name/display name pairs that will be used to create the list. Valid values can also contain a name which will be the actual value of the column and a display value which allows the Control Center to display use name.
Name	The name of the column in the database.
Display Name	Readable display name.
Validation	

Field	Description
Regular Expression	The Java regular expression that will be run on the proposed new cell value to validate it.
Regular Expression Description	A message to the user indicating what the regular expression is trying to check.
Runtime Binding	Runtime binding is how key column data gets filled out (bound) from data in the current session. There are multiple ways to bind this data and it is also possible to set an operator to define what should match (equals, less than, etc).
None	
Bind to Subscriber AVP	This pulls the value from an AVP on the subscriber. It will also pull values from a session AVP or a Policy Derived AVP.
Bind to Session/Policy State	This pulls the value from a Policy State Data Retriever which knows how to retrieve a single value for a session.
Bind to a result column from another table	This allows the key to be filled out from a columns value from another table. This allows 'normalizing' the table structure and not having on giant table with a lot of duplicated values.
Bind to Diameter request AVP code	This allows the key be filled out from an AVP on the diameter request.
Matching Operator	This allows the row to be 'matched' in other ways than having the value be 'equals'. Default value is equals. • eq: Equal • ne: Not Equal • gt: Greater than • gte: Greater than or equal • lt: Less than • lte: Less than or equal

Architecture

MongoDB Caching

The MongoDB database containing the CRD tables and the data is located in the MongoDB instance specified in the CRD plugin configuration.

The database is named cust_ref_data.

Two system collections exist in that database and do not actually contain CRD data:

- system.indexes - It is used by MongoDB. These are indices set on the database.
- crdversion - It contains a document indicating the version of all the CRD tables you have defined. The version field increments by one every time you make a change or add data to any of the CRD tables.

A collection is created for each CRD table defined in Policy Builder.

- This collection contains a document for each row you define in the CRD table.
- Each document contains a field for each column you define in the CRD table.
- The field contains the value specified for the column for that row in the table.
- Additionally, there is a _id field which contains the internal key used by MongoDB and _version which is used by CPS to provide optimistic locking protection, essentially to avoid two threads overwriting the other's update, on the document.

An example is shown below:

Figure 9: CRD Table In Policy Builder

Setting the Cache Results to true (checked) is the default and recommended settings in most cases as it yields the best performance. Use of the cached copy also removes the dependency on the availability of the CRD database. So if there is an outage or performance issue the policy decisions utilizing the CRD data will not be impacted.

The cached copy of the table is refreshed on CPS restart and whenever the API writes a change to the CRD table, otherwise the cached copy is used and the database is not accessed.

API Endpoints And Examples

The URL used to access the CRD API is located at https://<masterip or control ip>/custrefdata/<tablename>/_<operation>

Query API

Purpose

Returns all rows currently defined in the specified table.

HTTP Operation Type

GET

Example URL

https://<master or control ip>:8443/custrefdata/test/_query

Example URL with Filtering

https://<master or control ip>:8443/custrefdata/test/_query?key1=Platinum

Payload

None, although parameters can be specified on the URL for filtering.

Response

Success returns code 200 Ok; XML indicating rows defined is returned. If there are no records in the table, 200 Ok is returned with empty rows in it.

If the table does not exist, code 400 Bad Request is returned.

Example Response without Filtering

```
<rows>
  <row>
    <field code=”field1” value=”1004”/>
    <field code=”field2” value=”testee”/>
    <field code=”key1” value=”Platinum”/>
  </row>
  <row>
    <field code=”field1” value=”1004”/>
    <field code=”field2” value=”testee”/>
    <field code=”key1” value=”Platinum99”/>
  </row>
  <row>
    <field code=”field1” value=”field1example1”/>
    <field code=”field2” value=”field2example1”/>
    <field code=”key1” value=”key1example1”/>
  </row>
  <row>
    <field code=”field1” value=”field1example2”/>
    <field code=”field2” value=”field2example2”/>
    <field code=”key1” value=”key1example2”/>
  </row>
</rows>
```

Example Response with Filtering

```
<rows>
<rows>
  <row>
    <field code=”field1” value=”1004”/>
    <field code=”field2” value=”testee”/>
    <field code=”key1” value=”Platinum”/>
  </row>
</rows>
```

The response returns keys with the tag “field code”. If you want to use the output of Query as input to one of the other APIs, the tag needs to be changed to “key code”. Currently using “field code” for a key returns code 404 Bad Request and a java.lang.NullPointerException.

Create API

Purpose

Create a new row in the specified table.

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/test/_create

Example Payload

```
<row>
  <key code=”key1” value=”Platinum”/>
  <field code=”field1” value=”1004”/>
  <field code=”field2” value=”testee”/>
</row>
```

Response

Success returns code 200 Ok; no data is returned. The key cannot already exist for another row; submission of a duplicate key returns code 400 Bad Request.

If creating a row fails, API returns 400 Bad Request.

Note Create API does not support SVN CRD table operations and displays the following error message when Snv Crd Data checkbox is enabled in CRD table configuration:

Create operation is not allowed for subversion table

Update API

Purpose

Updates the row indicated by the key code in the table with the values specified for the field codes.

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/test/_update

Example Payload

```
<row>
  <key code="key1" value="Platinum"/>
  <field code="field1" value="1005"/>
  <field code="field2" value="tester"/>
</row>
```

Response

Success returns code 200 Ok; no data is returned. The key cannot be changed. Any attempt to change the key returns code 404 Not Found.

If updating a row fails, API returns 400 Bad Request.

Note Update API does not support SVN CRD table operations and displays the following error message when Snv Crd Data checkbox is enabled in CRD table configuration:

Update operation is not allowed for subversion table

Delete API

Purpose

Removes the row indicated by the key code from the table.

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/test/_delete

Example Payload

```
<row>
<key code="key1" value="Platinum"/>"/>
</row>
```

Response

Success returns code 200 Ok; no data is returned. If the row to delete does not exist, code 404 Not Found is returned.

If deleting a row fails, API returns 400 Bad Request.

Note Delete API does not support SVN CRD table operations and displays the following error message when Snv Crd Data checkbox is enabled in CRD table configuration:

Delete operation is not allowed for subversion table

Data Comparison API

Purpose

Determines whether the same CRD table data content is being used at different data centers.

The following three optional parameters can be provided to the API:

- tableName: Returns the checksum of a specified CRD table `tableName` indicating if there is any change in the specified table. If the value returned is same on different servers, it means there is no change in the configuration and content of that table.
- includeCrdversion: Total database checksum contains combination of checksum of all CRD tables configured in Policy Builder. If this parameter is passed as true in API, then total database checksum includes the checksum of "crdversion" table. Default value is false.
- orderSensitive: Calculates checksum of the table by utilizing the order of the CRD table content. By default, it does not sort the row checksums of the table and returns order sensitive checksum of every CRD table. Default value is true.

custrefdata/_checksum

Database level Checksum API returns checksum details for all the CRD tables and the database. If the value returned is same on different servers, there will be no change in the configuration and content of any CRD table configured in Policy Builder.

HTTP Operation Type

GET

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_checksum

Response

```
<response>
    <checksum><all-tables-checksum></checksum>
    <tables>
        <table name="<table-1-name>" checksum="<checksum-of-table-1>"/>
        <table name="<table-2-name>" checksum="<checksum-of-table-2>"/>

        <table name="<table-n-name>" checksum="<checksum-of-table-n>"/>
   </tables>
</response>
```

/custrefdata/_checksum?tableName=<user-provided-table-name>

Table specific Checksum API returns the checksum details for the specific CRD table. If the value returned is same on different servers, there will be no change in the configuration and content of that table.

HTTP Operation Type

GET

Example Endpoint URL

https://<master or control ip>:8443 /custrefdata/_checksum?tableName=<user-provided-table-name>

Response

```
<response>
    <tables>
        <table name="<user-provided-table-name>" checksum="<checksum-of-specified-table"/>
    </tables>
</response>
```

Note Table specific Checksum API does not support SVN CRD table operations and displays the following error message when Snv Crd Data checkbox is enabled in CRD table configuration:

Checksum operation is not allowed for subversion table

Table Drop API

Purpose

Drops custom reference table from MongoDB to avoid multiple stale tables in the system.

The Table Drop API is used in the following scenarios:

- If a CRD table does not exist in Policy Builder but exists in the database, the API can be used to delete the table from the database.
- If a CRD table exists in Policy Builder and database, the API cannot delete the table from the database. If this is attempted the API will return an error: “Not permitted to drop this table as it exists in Policy Builder”.
- If a CRD table does not exist in Policy Builder and database, the API will also return an error `No table found:<tablename>`.

/custrefdata/<table_name>/_drop

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/<table_name>/_drop

Note Drop API does not support SVN CRD table operations and displays the following error message when Snv Crd Data checkbox is enabled in CRD table configuration:

Drop operation is not allowed for subversion table

Export API

Purpose

Exports single and multiple CRD table and its data.

/custrefdata/_export?tableName=<table_name>

Exports single CRD table and its data.

Returns an archived file containing csv file with information of specified CRD table `table_name`.

HTTP Operation Type

GET

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_export?tableName=<table_name>

/custrefdata/_export

Exports all CRD tables and its data.

Returns an archived file containing csv file with information for each CRD Table.

HTTP Operation Type

GET

Example Endpoint URL

https://<master or control ip>:8443 /custrefdata/_export

Note Export API does not support Svn CRD tables and displays the following warning message in the Response Header "Export-Warning":

Datasource for tables [table1, table2,...] is subversion. Response will not contain data for these tables and skipped SVN CRD tables to be a part of archive.

Import API

Purpose

Imports CRD table and its data.

It takes an archived file as an input which contains one or more csv files containing CRD tables information.

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_import

https://<lbvip01>:8443/custrefdata/_import?batchOperation=true

https://<lbvip01>:8443/custrefdata/_import?batchOperation=false&duplicateValidation=true

Note

1. The "batchOperation" flag is used to insert CRD data in the batch. The default value is true and if you do not provide it in the request parameter the default value is taken.
2. The "duplicateValidation" flag is used to validate or invalidate duplicate data in the archive. The default value is true and if you do not provide it in the request parameter the default value is taken which means it will always validate your data as duplicate.
3. If "batchOperation" is true, the API will validate your data as duplicate data regardless of the value provided for "duplicateValidation".

Note Import API supports SVN CRD table operations in the following scenarios:

- If the archive contains only mongodb tables, success message is displayed in the response.
- If the archive contains only SVN tables, success and warning messages are displayed in the response.
- If the archive contains both mongodb and SVN tables, success and warning messages are displayed in the response.

Snapshot POST API

Purpose

Creates a snapshot of the CRD tables on the system. The created snapshot will contain CRD table data, policy configuration and checksum information for all CRD tables.

/custrefdata/_snapshot?userId=<user_id>&userComments=<user_comments>

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_snapshot?userId=<user_id>&userComments=<user_comments>

Optional Parameters

userComments

Note Snapshot POST API does not support export of the contents of Svn CRD tables. The API returns the following warning message if there are any Svn CRD tables present while creating snapshot:

Datasource for tables [table_1, table_2…] is subversion. Data for these tables will not come from database (mongodb)

Snapshot GET API

Purpose

Enables you to get the list of all valid snapshots in the system.

The following information is available in the list of snapshots:

- Snapshot name
- Snapshot path
- Date and time of snapshot creation
- User comments provided on creation of the snapshot
- Checksum information of CRD tables
- Policy configuration SVN version number

/custrefdata/_snapshot

HTTP Operation Type

GET

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_snapshot

Example Response

```
<snapshots>
    <snapshot>
        <name><date-and-time>_<user-id></name>
        <snapshotPath>/var/broadhop/snapshot/20160620011825306_qns</snapshotPath>
        <creationDateAndTime>20/06/2016 01:18:25:306</creationDateAndTime>
        <comments>snapshot-1 june</comments>
        <policyVersion>903</policyVersion>
        <checksum checksum="60f51dfd4cd4554910da44a776c66db1">
            <table name=<table-name-1> checksum="<table-checksum-1>"/>
                …
            <table name=<table-name-n> checksum="<table-checksum-n>"/>
        </checksum>
    </snapshot>
    <snapshot>
                …
```

```
    </snapshot>
</snapshots>
```

Note Snapshot GET API does not return checksum information of Svn CRD tables as they are not part of created snapshots.

Revert API

Purpose

Enables you to revert the CRD data to a specific snapshot. If the specific snapshot name is not provided, the API will revert to the latest snapshot.

/custrefdata/_revert?snapshotName=<snapshot_name>

HTTP Operation Type

POST

Example Endpoint URL

https://<master or control ip>:8443/custrefdata/_revert?snapshotName=<snapshot_name>

Optional Parameter

snapshotName

Note Revert API does not support reverting of CRD data for Svn CRD tables. For Svn CRD table, it clears the mongodb table and displays the following warning message:

Datasource for tables [table_1, table_2…] is subversion. Data for these tables will be reverted using svn datasource not from database (mongodb)

Tips for Usage

The Query API is a GET operation which is the default operation that occurs when entering a URL into a typical web browser.

The POST operations, Create, Update, and Delete, require the use of a REST client so that the payload and content type can be specified in addition to the URL. REST clients are available for most web browsers as plug-ins or as part of web service tools, such as SoapUI. The content type when using these clients should be specified as application/xml or the equivalent in the chosen tool.

View Logs

You can view the API logs with the following commands:

- monitor log application – tail the current application log
- monitor log engine – tail the current engine log
- monitor log container – tail a specific container log
- show log application - view the current application log
- show log engine – view the current engine log

Logging Support Using Journald

To monitor and view logs, journald system service has been added that collects and stores logging data. It creates and maintains structured, indexed journals based on logging information received from a variety of sources. The following is a sample of CLI commands:

- `monitor log application` - This command is used to tail the current Policy Server (qns) log.
- `monitor log engine` - This command is used to tail the current Policy Server (qns) engine log
- `monitor log container <container id>` - This command is used to tail the container logs.
- `show log application` - This command opens the consolidated logs.
- `show log engine` - This command is used to open the consolidate engine logs using Linux 'less' command.

For further log access, you need to connect to the OpenStack control node and from there to respective master or control node. For example, to connect to master/control nodes use the following command:

```
ssh -i cps.pem cps@IPAddress
```

where, *IPAddress* is the IP address of the master or control node.

To access the logs once you are connected to control node, use the following command:

```
docker logs container-id
```

For example, use `docker logs mongo-s1` to display all the logs of mongo-s1 container.

Bulk Provisioning of Records in SLF Database

CPS vDRA provides APIs for bulk provisioning of subscriber records in the SLF database.

You can use the CSV file to provision create and update of bulk subscriber records using SLF API. You can also check the status of the upload using the API.

Note SLF bulk provisioning generates high number of database write operations in a short duration of time. To spread out the operations over a period of time and mitigate the performance issue, configure the transactions per second (TPS) for SLF provisioning in Policy Builder.

For more information, see the *CPS vDRA Configuration Guide*.

CSV File

The CSV file format is used to bulk provision the subscriber records in SLF database. The Actions column in the CSV file determines whether the record is for creation, updation, or deletion.

You can use # in the beginning of the line to indicate comments in the CSV file. The line is ignored when the file is processed.

Table 4: CSV File Format

Column	Description
Action	The action to be performed on the subscriber record. • Create - creates subscriber record if it does not exist. • Put – creates the subscriber record, if it does not exist; if subscriber record already exists, updates the subscriber record. • Delete – deletes the subscriber record, if it exists.
Subscriber Id	The subscriber ID of the subscriber.
IMSI	The IMSI of the subscriber. If the same subscriber has multiple IMSI, then add multiple IMSI columns for the subscriber.
MSISDN	The MSISDN of the subscriber. If the same subscriber has multiple MSISDN, then add multiple MSISDN columns for the subscriber.
Destination:<Tag>	The destinations of the subscriber. To provision multiple destinations, add column name/header with prefix “Destination:” and suffix it with the tag, for example: Destination:HSS, Destination:MME, Destination:PCRF, etc

Sample CSV File

```
Action, Subscriber Id, IMSI, IMSI, MSISDN, MSISDN, Destination:MME, Destination:HSS
Put,    1001,         34101, 34102, 91001, 91002,  MME1,            HSS1
Put,    1001,         34101,      ,  91005,     ,  MME2,            HSS2
Delete, 1010, , , , , ,
```

Bulk Upload API

Schedules the SLF bulk subscribers provisioning task. Bulk Upload API takes the input as csv file and schedules the job to execute in the background.

Request

Method: POST

URI: /dra/slfapi/subscriber/bulkUpload

Header: Content-Type: multipart/form-data

Body: CSV File

Request Example

```
HTTP POST /dra/slfapi/subscriber/bulkUpload
```

Response Example

```
HTTP STATUS: 202 (Accepted)
{
  "success": {
    "code": 1,
    "message": "Request accepted, slf bulk upload task is scheduled for execution"
  }
}
```

Example of Curl Command

```
curl -X POST --progress-bar -H "Content-Type: multipart/form-data"
-H "Content-Type: application/json" \ -F "file=@create_subscribers.csv"
https://<MasterIP>/dra/slfapi/subscriber/bulkUpload --insecure
-u admin:admin
```

The file named create_subscribers.csv must be created before running this command.

Bulk Upload Status

Returns the list of bulk upload status of the bulk provisioning sorted by the latest first. Latest 10 statuses would be saved in the system for reference, old status will automatically get purged.

The following table describes the fields in the Bulk Upload Status:

Table 5: Bulk Upload Status

Field	Description
fileName	The name of csv file uploaded.
startTime	The time when task was scheduled.
endTime	The time when task was finished
approxEndTime	The future time when task is expected to be finished
status	The status of the task Status can be one of these statues (scheduled, in-progress, complete, failed)
statusMessage	The detailed status of the task

Field	Description
numberOfTotalSubscriber	Total number of subscriber in csv file
numberOfPending	The number of subscriber pending for execution
numberOfComplete	The number of subscriber, whose execution is finished
numberOfSuccess	The number of subscriber provisioned successfully.
numberOfFailure	The number of subscriber failed in provisioning.
failedSubscriber	This field contains the failure reason for each failed subscriber. This is a map, with key as error code and value as the list of failed subscribers.

Request

Method: GET

URI: /dra/slfapi/subscriber/bulkUploadStatus

Request Example

```
HTTP GET /dra/slfapi/subscriber/bulkUploadStatus
```

Response Example

```
HTTP STATUS: 200
 [{
 "approxEndTime": "08-17-2017 13:31:59",
        "failedSubscriber": {
            "1001": [
                "1000000000",
                "1000000001",
                "1000000002"
            ]
         },
         "fileName": "create_subscribers_1k.csv",
         "numberOfComplete": 700,
         "numberOfFailure": 3,
         "numberOfPending": 300,
         "numberOfSuccess": 697,
         "numberOfTotalSubscriber": 1000,
         "startTime": "08-17-2017 13:30:16",
         "status": "complete",
         "statusMessage": "Slf bulk upload task execution is in progress"
    },
   {
    "endTime": "08-18-2017 12:41:27",
         "failedSubscriber": {},
     "fileName": "create_subscribers_10.csv",
         "numberOfComplete": 10,
         "numberOfFailure": 0,
         "numberOfPending": 0,
     "numberOfSuccess": 10,
         "numberOfTotalSubscriber": 10,
         "startTime": "08-18-2017 12:41:27",
     "status": "complete",
```

```
        "statusMessage": "Slf bulk upload task is completed"
    }]
```

Example of Curl Command

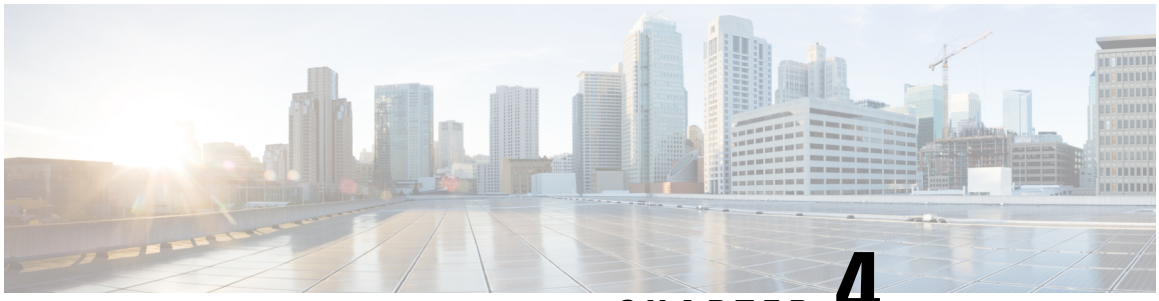
```
curl -X GET --progress-bar -H "Content-Type:
application/json" \https://<MasterIP>/dra/slfapi/subscriber/bulkUploadStatus
 --insecure -u admin:admin
```

vDRA Peer API

The vDRA Peer API provides a REST API interface for the following functions:

- view active and inactive peer endpoints - local and remote
- view peer details for each host and/or peer key
- peer status logs

For more information about the Peer API, see the API RAML at: https://<master ip>/dra/docs/api/



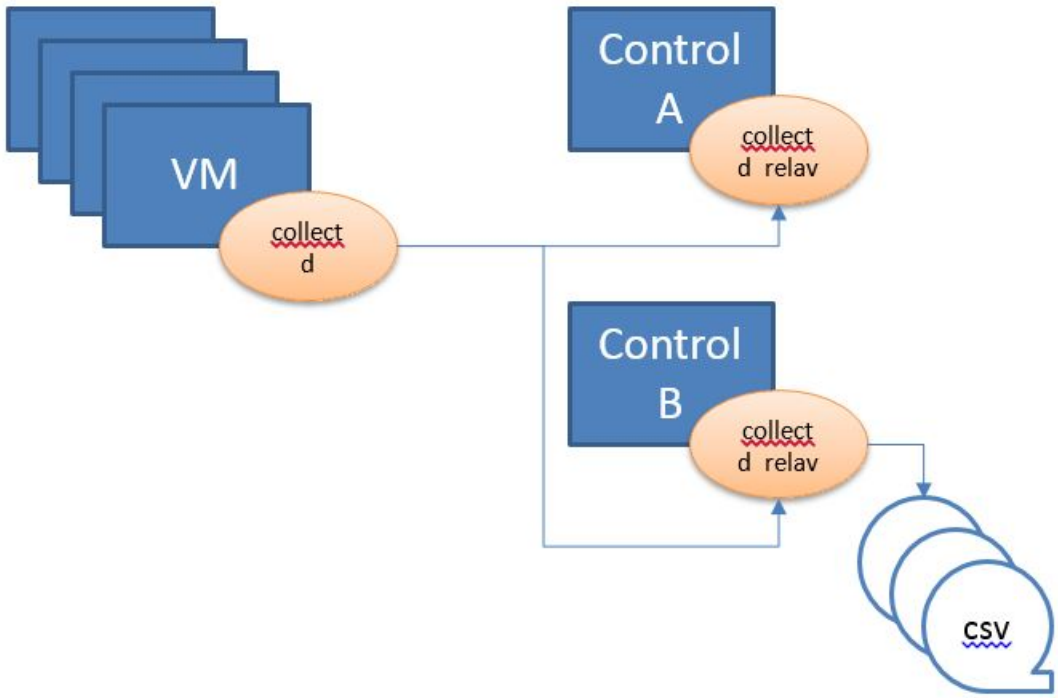
CHAPTER 4

CPS Statistics

Bulk Statistics Overview

Bulk Statistics are the statistics that are gathered over a given time period and written to a set of CSV files. These statistics can be used by external analytic processes and/or network management systems. The architecture of CPS bulk statistic collection is shown in the following illustration.

Figure 10: Architecture of CPS Bulk Statistic Collection



The collection utility collectd is used for collecting and storing statistics from each VM. Detailed collectd documentation can be found on http://collectd.org/.

Collectd within CPS is deployed with nodes relaying data using the collectd network plug-in (https://collectd.org/wiki/index.php/Plugin:Network) to the centralized collection nodes on the control-A and control-B virtual machines. The centralized collector writes the collected data to output CSV files.

Note Control A and Control B collect bulk statistics independently. As a result, it is normal to have slight differences between the two files. For example, control-A will generate a file at time t and control-B will generate a file at time t +/- the clock drift between the two machines.

As a best practice, always use the bulk statistics collected from Control-A. Control-B can be used as a backup in the event of failure of control-A.

In the event that Control-A becomes unavailable, statistics will still be gathered on Control-B. Statistics data is not synchronized between Control-A and Control-B, so a gap would exist in the collected statistics while control-A is down.

Note The collectd collection mechanism are separate from the Prometheus / Grafana Monitoring.

CPS Statistics

The list of statistics available in CPS is consolidated in an Excel spreadsheet. After CPS is installed, this spreadsheet can be found in the following location:

https://<masterip>/dra/docs/stats/bulk-stats.html

Bulk Statistics Collection

By default, CPS outputs a bulk statistics CSV file to the `/var/broadhop/stats/` directory on the control-A and control-B VMs in five-minute intervals.

An scp / sftp daemon running on port 2026 retrieves all statistics within the `/var/broadhop/stats` directory. Only locally defined users within the scheduling application associated to the “bulkstats” or “admin” group are able to retrieve statistics.

You can also retrieve statistics by logging into the virtual machine directly and retrieving the statistics from the `/data/stats` directory.

The default naming standard is `bulk-hostname-YYYY-MM-DD-HH-MI.csv`

These CSV files include all statistics collected from all VMs during the five-minute interval.

Note If a statistic is generated by the system multiple times within the five-minute interval, only the last measured statistics is collected in the CSV file.

The following list is a sample of the file names created in the `/var/broadhop/stats/` directory on the control-A VM:

```
[root@control-1 stats]# pwd
/data/stats-relay-s1/var/broadhop/stats [root@control-A stats]# ls
bulk-control-A-201510131350.csv
bulk-control-A-201510131355.csv
bulk-control-A-201510131400.csv
bulk-control-A-201510131405.csv
bulk-control-A-201510131410.csv
bulk-control-A-201510131415.csv
bulk-control-A-201510131420.csv
bulk-control-A-201510131425.csv
bulk-control-A-201510131430.csv
bulk-control-A-201510131435.csv
bulk-control-A-201510131440.csv
bulk-control-A-201510131445.csv
bulk-control-A-201510131450.csv
bulk-control-A-201510131455.csv
bulk-control-A-201510131500.csv
bulk-control-A-201510131505.csv
bulk-control-A-201510131510.csv
bulk-control-A-201510131515.csv
bulk-control-A-201510131520.csv
bulk-control-A-201510131525.csv
bulk-control-A-201510131530.csv
bulk-control-A-201510131535.csv
bulk-control-A-201510131540.csv
bulk-control-A-201510131545.csv
bulk-control-A-201510131550.csv
bulk-control-A-201510131555.csv
bulk-control-A-201510131600.csv
bulk-control-A-201510131605.csv
bulk-control-A-201510131610.csv
bulk-control-A-201510131615.csv
bulk-control-A-201510131620.csv
bulk-control-A-201510131625.csv
bulk-control-A-201510131630.csv
```

Retention of CSV Files

CPS retains each bulk statistics CSV file on the control-A/B VM for two days; after which the file is automatically removed.

If you need to preserve these CSV files, you must back up the files or move them to an alternate system.

Diameter Monitoring KPIs

The following table describes CPS KPIs that are useful for monitoring Diameter message traffic.

Note As each deployment is unique, no recommended ranges are provided. Cisco recommends monitoring these KPIs for a period of time (1-3 months) to establish a baseline. Deviations can then be monitored from the baseline values.

Table 6: Diameter Monitoring KPIs

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Gx/F	Diameter Round Trip	node[x].messages.e2e_<domain>_ [realm_] Gx_CCR-I_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages.e2e_<domain>_ [realm_] Gx_CCR-I_2001. qns_stat.total _time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages.e2e _<domain>_[realm_] Gx_CCR-I_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-I_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-I_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/A	Diameter Input Queue	node1.counters. [realm_] Gx_CCR-I.qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director

Appid/Monitoring Area	Category	Statistic	Description	Availability/Node
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/A	Diameter Input Queue	node1.counters. [realm_] Gx_CCR-U. qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_2001. qns_stat. total_time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-U_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/A	Diameter Input Queue	node1.counters. [realm_] Gx_CCR-U. qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-T_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-T_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-T_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-T_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_CCR-T_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/A	Diameter Input Queue	node1.counters. [realm_] Gx_CCR-T.qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director

AppId/Monitoring Area	Category	Statistic	Description	Availability/Node
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_timeout. qns_stat.success	Success timeout count for RAR message	Policy Director
Gx/A	Diameter Input Queue	node1.counters. [realm_] Gx_RAA.qns_count	Count of all messages sent to the policy engine	Policy Server (qns)
Gx/A	Diameter Input Queue	node1.messages. in_q_Gx_RAA. qns_stat.error	Count of messages failed to be sent to the policy engine	Policy Server (qns)
Gx/A	Diameter Input Queue	node1.messages. in_q_Gx_RAA. qns_stat.success	Count of messages successfully sent to the policy engine	Policy Server (qns)
Gx/E	Diameter Output Queue	node1.counters. [realm_] Gx_RAR.qns_count	Count of messages successful sent to the Policy Director (LB)	Policy Server (qns)
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_timeout. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_RAA.qns_count	Count of messages successful sent to the Policy Director (LB)	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_AAR_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Rx/E	Diameter Output Queue	node1.counters. [realm_] Rx_AAA_2001. qns_count	Count of AAA messages with result-code = 2001 sent successfully to the PolicyDirector (LB)	Policy Server (qns)
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director

Appid/Monitoring Area	Category	Statistic	Description	Availability/Node
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_retry. qns_count	Retry count for ASR message	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_ASA_bypass. qns_count	Count of message that do not require processing by the policy engine	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_]Rx_ASA. qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_ASA_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_RAR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_RAR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_RAR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_RAR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Rx/F	Diameter Input Queue	node[x].messages. e2e_<domain>_ [realm_] Rx_RAR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/A	Diameter Input Queue	node1.counters.[realm_] Rx_RAA_bypass. qns_count	Count of message that do not require processing by the policy engine	Policy Server (qns)
Rx/A	Diameter Output Queue	node1.counters.[realm_] Rx_RAA.qns_count	Count of messages successfully sent to the policy engine	Policy Server (qns)

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Rx/A	Diameter Round Trip	node1.counters.[realm_] Rx_RAA_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_STR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_STR_2001. qns_stat.total_time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_STR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_STR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_STR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/A	Diameter Input Queue	node1.counters.[realm_] Rx_STR.qns_count	Count of messages successful sent to the policy engine	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_STR_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.messages. in_q_Rx_STR. qns_stat.success	Count of messages successful sent to the policy engine	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.messages. in_q_Rx_STR. qns_stat. total_time_in_ms	Total milliseconds of messages successfully sent to the policy engine	Policy Server (qns)

Appid/Monitoring Area	Category	Statistic	Description	Availability/Node
Rx/D	Engine Message	node1.messages. diameter_Rx_STR. qns_stat.success	Success message count	Policy Server (qns)
Rx/D	Engine Message	node1.messages. diameter_Rx_STR. qns_stat. total_time_in_ms	Total milliseconds of successful messages	Policy Server (qns)
Rx/E	Diameter Input Queue	node1.counters. [realm_] Rx_STA_2001. qns_count	Count of STA messages with result-code = 2001 sent successfully to the PolicyDirector (LB)	Policy Server (qns)
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_2001. qns_stat. total_time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_SLR_bypass. qns_count	Count of message that do not require processing by the policy engine	Policy Server (qns)

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Sy/A	Diameter Input Queue	node1.counters.[realm_] Sy_SLR.qns_count	Count of messages successful sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.counters.[realm_] Sy_SLR_drop.qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages. in_q_Sy_SLA. qns_stat.success	Count of messages successfully sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages.in_q_Sy_SLA. qns_stat. total_time_in_ms	Total milliseconds of messages successfully sent to the policy engine	Policy Server (qns)
Sy/D	Engine Message	node1.messages. diameter_Sy_SLA. qns_stat.success	Success message count	Policy Server (qns)
Sy/D	Engine Message	node1.messages. diameter_Sy_SLA. qns_stat. total_time_in_ms	Total milliseconds of successful messages	Policy Server (qns)
Sy/B	Diameter Action	node1.actions. send.diameter_ Sy_SLR.qns_stat.success	Success actions count	Policy Server (qns)
Sy/B	Diameter Action	node1.actions. send.diameter_ Sy_SLR.qns_stat. total_time_in_ms	Total milliseconds of successful messages	Policy Server (qns)
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SNR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SNR_2001. qns_stat. total_time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SNR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SNR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SNR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_SNR.qns_count	Count of messages successful sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_SNR_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages. in_q_ Sy_SNR. qns_stat.success	Count of messages successfully sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages. in_q_Sy_SNR. qns_stat. total_time_in_ms	Total milliseconds of messages successfully sent to the policy engine	Policy Server (qns)
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_STR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_STR_2001. qns_stat. total_time_in_ms	Total milliseconds of successful messages with return code matching2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_STR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director

Appld/Monitoring Area	Category	Statistic	Description	Availability/Node
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_STR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_STR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_STA_bypass. qns_count	Count of message that do not require processing by the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_STA.qns_count	Count of messages successful sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.counters. [realm_] Sy_STA_drop. qns_count	Count of messages dropped due to exceedingSLA	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages. in_q_Sy_STA. qns_stat.success	Count of messages successfully sent to the policy engine	Policy Server (qns)
Sy/A	Diameter Input Queue	node1.messages. in_q_Sy_STA. qns_stat.total_ time_in_ms	Total milliseconds of messages successfully sent to the policy engine	Policy Server (qns)
Sy/D	Engine Message	node1.messages. diameter_Sy_STA. qns_stat.success	Success message count	Policy Server (qns)
Sy/D	Engine Message	node1.messages. diameter_Sy_STA. qns_stat.total_time_in_ms	Total milliseconds of successful messages	Policy Server (qns)
Sy/B	Diameter Action	node1.actions.send. diameter_Sy_STR. qns_stat.success	Success actions count	Policy Server (qns)
Sy/B	Diameter Action	node1.actions.send. diameter_ Sy_STR.qns_stat. total_time_in_ms	Total milliseconds of successful actions	Policy Server (qns)

Appid/Monitoring Area	Category	Statistic	Description	Availability/Node
Sy/E	Diameter Output Queue	node1.counters. [realm_] Sy_STR.qns_count	Count of messages successfully sent to the Policy Director (LB)	Policy Server (qns)

Example CPS Statistics

Sample CSV Files

The following list is a sample of the file names created in the /var/broadhop/stats directory on the control-A VM.

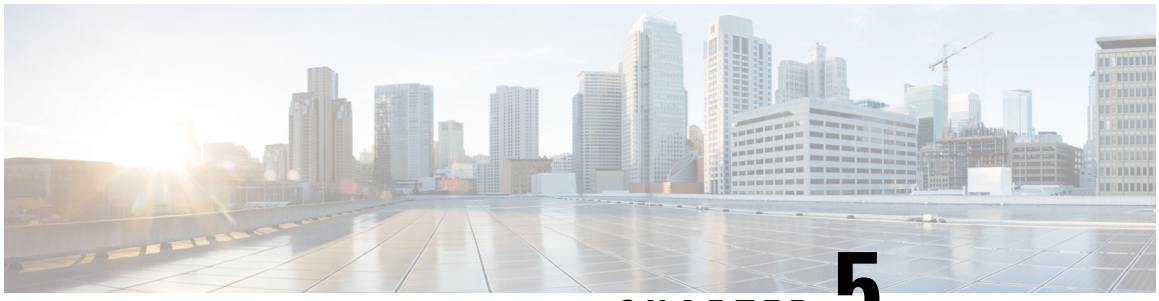
```
[root@control-A stats]# pwd
/var/broadhop/stats [root@control-A stats]# ls
bulk-control-A-201510131350.csv
bulk-control-A-201510131355.csv
bulk-control-A-201510131400.csv
bulk-control-A-201510131405.csv
bulk-control-A-201510131410.csv
bulk-control-A-201510131415.csv
bulk-control-A-201510131420.csv
bulk-control-A-201510131425.csv
bulk-control-A-201510131430.csv
bulk-control-A-201510131435.csv
bulk-control-A-201510131440.csv
bulk-control-A-201510131445.csv
bulk-control-A-201510131450.csv
bulk-control-A-201510131455.csv
bulk-control-A-201510131500.csv
bulk-control-A-201510131505.csv
bulk-control-A-201510131510.csv
bulk-control-A-201510131515.csv
bulk-control-A-201510131520.csv
bulk-control-A-201510131525.csv
bulk-control-A-201510131530.csv
bulk-control-A-201510131535.csv
bulk-control-A-201510131540.csv
bulk-control-A-201510131545.csv
bulk-control-A-201510131550.csv
bulk-control-A-201510131555.csv
bulk-control-A-201510131600.csv
bulk-control-A-201510131605.csv
bulk-control-A-201510131610.csv
bulk-control-A-201510131615.csv
bulk-control-A-201510131620.csv
bulk-control-A-201510131625.csv
bulk-control-A-201510131630.csv
```

Sample Output

C,<VM_name>,node1.actions.send.diameter_Gx_CCA-I.qns_stat.success,19 where the <VM_Name> indicates the VM where statistics has been collected.

A sample bulk statistics.csv file is shown below:

```
C,qns01,node1.actions.SaveSubscriberActionImpl.qns_stat.error,0
C,qns01,node1.actions.SaveSubscriberActionImpl.qns_stat.success,6
C,qns01,node1.actions.send.diameter_Gx_CCA-I.qns_stat.error,0
C,qns01,node1.actions.send.diameter_Gx_CCA-I.qns_stat.success,19
C,qns01,node1.actions.send.diameter_Gx_CCA-T.qns_stat.error,0
C,qns01,node1.actions.send.diameter_Gx_CCA-T.qns_stat.success,9
D,qns01,node1.messages.in_q_Gx_CCR-I.qns_stat.total_time_in_ms,14
D,qns01,node1.messages.in_q_Gx_CCR-T.qns_stat.total_time_in_ms,2
D,qns01,node1.messages.in_q_Gx_CCR-U.qns_stat.total_time_in_ms,1
D,qns01,node1.messages.in_q_Gx_RAA.qns_stat.total_time_in_ms,0
D,qns01,node1.messages.in_q_Sh_SNA.qns_stat.total_time_in_ms,2
D,qns01,node1.messages.in_q_Sh_UDA.qns_stat.total_time_in_ms,0
D,qns01,node1.messages.TimerExpired.qns_stat.total_time_in_ms,7244
D,qns01,node1.spr.createSubscriber.qns_stat.total_time_in_ms,29
D,qns01,node1.spr.deleteSubscriber.qns_stat.total_time_in_ms,40
D,qns01,node1.spr.getSubscriber.qns_stat.total_time_in_ms,44
D,qns01,node1.spr.updateSubscriber.qns_stat.total_time_in_ms,21
G,lb02,node1.ldap.SITELDAP.qns_ldap_connection.MaximumAvailableConnections,10.0
G,lb02,node1.ldap.SITELDAP.qns_ldap_connection.NumAvailableConnections,0.0
G,lb02,node1.thread.gauge.daemon_thread_count,80.0
G,lb02,node1.thread.gauge.live_thread_count,184.0
```



CHAPTER 5

CLI Commands

CLI Command Overview

The command-line interface (CLI) is one of the available user interfaces to configure and monitor the launched application. This user interface provides direct access to execute commands via remote access methods over SSH.

In addition to the CLI, Cisco CPS provides a NETCONF and RESTCONF interface for API access to the application.

CLI Command Modes

The CLI provides two separate command modes – OPERATIONAL and CONFIG.

Each command mode has a separate set of commands available for configuration and monitoring of the application. Entering a "?" at the command problem will indicate the list of available commands for execution within a given mode.

When you start a session, the default mode is OPERATIONAL mode. From this mode, you can access monitoring "show" commands, debugging commands and system maintenance commands. You can enter CONFIG mode to change configuration by issuing the "config" command at the OPERATIONAL prompt.

OPERATIONAL Mode

Logging into the master VM on port 2024 via SSH will allow you to access OPERATIONAL mode. The login into the system will require the use of a username and password. You may attempt to enter a correct password up to three times before the connection attempt is refused.

The commands available at the OPERTIONAL level are separate from the ones available at the CONFIG level. In general, the OPERATIONAL commands encompass monitoring, debugging, and maintenance activity a user will perform.

To list the available OPERATIONAL commands, use the following command:

Table 7: List Commands of OPERATIONAL Mode

Command	Purpose
scheduler# ?	Lists the user OPERATIONAL commands

Example:

```
scheduler# ?
Possible completions:
  aaa                    AAA management
  apply
  autowizard             Automatically query for mandatory elements
  cd                     Change working directory
  clear                  Clear parameter
  commit                 Confirm a pending commit
  compare                Compare running configuration to another configuration or a file
  complete-on-space      Enable/disable completion on space
  config                 Manipulate software configuration information
  db                     DB connection and monitoring
  debug                  Debug commands
  describe               Display transparent command information
  devtools               Enable/disable development tools
  display-level          Configure show command display level
  docker                 Docker Management
  exit                   Exit the management session
  file                   Perform file operations
  help                   Provide help information
  history                Configure history size
  id                     Show user id information
  idle-timeout           Configure idle timeout
  ignore-leading-space   Ignore leading whitespace (true/false)
  job                    Job operations
  logger                 Log level management
  logout                 Logout a user
  monitor                Application monitoring
  no                     Negate a command or set its defaults
  output-file            Copy output to file or terminal
  paginate               Paginate output from CLI commands
  prompt1                Set operational mode prompt
  prompt2                Set configure mode prompt
  pwd                    Display current mode path
  quit                   Exit the management session
  screen-length          Configure screen length
  screen-width           Configure screen width
  script                 Script actions
  send                   Send message to terminal of one or all users
  show                   Show information about the system
  show-defaults          Show default values when showing the configuration
  source                 File to source
  system                 System management
  terminal               Set terminal type
  timestamp              Enable/disable the display of timestamp
  who                    Display currently logged on users
  write                  Write configuration
scheduler#
```

The list of commands will vary based on the version of software installed.

CONFIG Mode

Within OPERATIONAL mode, you can enter CONFIG mode by issuing the “config” command. In general, the CONFIG commands modify the system configuration.

To enter CONFIG mode, use the following command:

Table 8: Enter CONFIG mode

Command	Purpose
scheduler# config	Enter CONFIG mode of the CLI

In CONFIG mode, the prompt changes to include a “(config)” at the end of the prompt.

Example:

```
scheduler# config
Entering configuration mode terminal
scheduler(config)#
```

To list the available CONFIG commands, use the following command:

Table 9: List commands in CONFIG mode

Command	Purpose
scheduler(config)# ?	List the user CONFIG commands

Example:

```
scheduler(config)# ?
Possible completions:
  aaa             AAA management
  alert           Alert status
  alias           Create command alias.
  binding         Binding DB connections
  control-plane   Cross data center control plane
  docker          Docker Management
  license         CPS License Management
  nacm            Access control
  ntp             NTP configuration
  scheduling      Service scheduling
  session         Global default CLI session parameters
  statistics      Application statistics
  system          System configuration
  user            User specific command aliases and default CLI session parameters
  webui           Web UI specific configuration
  ---
  abort           Abort configuration session
  annotate        Add a comment to a statement
  clear           Remove all configuration changes
  commit          Commit current set of changes
  compare         Compare configuration
  copy            Copy a list entry
  describe        Display transparent command information
  do              Run an operational-mode command
  end             Terminate configuration session
  exit            Exit from current mode
  help            Provide help information
  insert          Insert a parameter
```

```
load            Load configuration from an ASCII file
move            Move a parameter
no              Negate a command or set its defaults
pwd             Display current mode path
rename          Rename an identifier
resolved        Conflicts have been resolved
revert          Copy configuration from running
rollback        Roll back database to last committed version
save            Save configuration to an ASCII file
service         Modify use of network based services
show            Show a parameter
tag             Manipulate statement tags
top             Exit to top level and optionally run command
validate        Validate current configuration
```

alert rule

Creates a new alerting rule.

The alerting rule allows automatic creation of internal and SNMP traps based on system conditions. The Prometheus monitoring application must be running for alerts to trigger properly. If all Prometheus servers are down, then the system does not generate alerts.

Syntax

```
alert rule name duration duration event-host-label event-host-label
expression expression message message snmp-clear-message snmp-clear-message
snmp-facility { application | hardware | networking | os | proc | virtualization }
snmp-severity { alert | critical | debug | emergency | error | info | none | notice | warning
 }
```

Command Parameters

Table 10: Parameter Description

Command Parameter	Description
name	The name of the alert rule.
duration	The duration measured the condition must exist before triggering an alarm. The format of the duration is <value><unit>. The value is any positive integer and the unit is one of the following: • s – second • m – minute • h – hour

Command Parameter	Description
event-host-label (optional)	The label received by the alerting engine from the Prometheus monitoring application. The application generates one alert per unique value of the given label. The valid labels are determined by the query executed and can be found by executing the query without the comparison operators in the Grafana application on a sample dashboard. If not defined, then the alert is considered global.
expression	The expression that makes up the alerting rule. The expression is built using a Prometheus expressions (https://prometheus.io/docs/querying/basics/) and must conform to the rules defined in the Prometheus alerting documentation: https://prometheus.io/docs/alerting/rules/
message	A configurable message to be sent with the alert. This message supports substitution of labels as defined in the templating section of the Prometheus documentation: https://prometheus.io/docs/alerting/rules/ . The resultant alert message is sent in any associated SNMP traps when the alert is triggered.
snmp-clear-message (optional)	A configurable message that is sent as the clear message when the alert condition is no longer valid.
snmp-facility (optional)	The target snmp-facility to use when generating SNMP trap: • application • hardware • networking • os • proc • virtualization Default is application.

Command Parameter	Description
snmp-severity	The target snmp-severity to use when generating an SNMP trap: • alert • critical • debug • emergency • error • info • none • notice • warning Default is alert.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the alert rule command to define monitoring rules for the system. When you create a new alert rule, the alert rule is exported to the Prometheus monitoring servers, which are monitoring the system on a 1-second interval. The Prometheus servers monitor the underlying expression defined in the alert rule and send alerts scheduling OAM node when they are triggered or when they are cleared. The OAM node tracks internally the status of all alerts and sends any SNMP traps if SNMP servers are defined.

Examples

The following example generates an alert when node_lode > 3:

```
alert rule test
 expression         "node_load5 > 3"
 event-host-label   instance
 message            "Node level exceeds 3"
 snmp-facility      application
 snmp-clear-message "Node level below 3"
!
```

alert snmp-v2-destination

Creates a new SNMPv2 destination.

Creation of a SNMPv2 destination causes the system to forward any triggered/cleared alerts to the SNMPv2 destination.

Syntax

```
alert snmp-v2-destination nms-address community community
```

Command Parameters

Table 11: Parameter Description

Command Parameter	Description
nms-address	The address to send SNMPv2 traps.
Community	The community to use for SNMPv2 traps

Command Mode

CONFIG

VNFs

All

Command Usage

Use the alert snmp-v2-destination to forward alerts from the system to an external SNMPv2 trap receiver. The traps are sent using the following algorithm:

- Sent once when the alert is cleared
- Sent once when the alert is firing
- Sent once if the OAM application is restarted and the alert is firing.

Examples

The following example sends all alerts to community “test” with address 10.10.10.10.

```
scheduler(config)# alert snmp-v2-destination 10.10.10.10 community test
```

alert snmp-v3-destination

Creates a new SNMPv3 destination.

Creation of a SNMPv3 destination causes the system to forward any triggered/cleared alerts to the SNMPv3 destination.

Syntax

```
alert snmp-v3-destination nms-address auth-password
auth-password auth-proto auth-proto engine-id
```

```
engine-id privacy-password privacy-password
user user
```

Command Parameters

Table 12: Parameter Description

Command Parameter	Description
nms-address	The address to send SNMPv3 traps.
auth-password	Authentication passphrase used for authenticated SNMPv3 messages.
auth-proto	Authentication protocol used for authenticated SNMPv3 messages. Valid values are MD5 and SHA
engine-id	Context engine id as a hexadecimal string.
privacy-password	Privacy passphrase used for encrypted SNMPv3 messages.
privacy-protocol	Privacy protocol used for encrypted SNMPv3 messages. Valid values are DES and AES.
user	Security name used for authenticated SNMPv3 messages.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the alert snmp-v3-destination to forward alerts from the system to an external SNMPv2 trap receiver. The traps are sent using the following algorithm:

- Sent once when the alert is cleared
- Sent once when the alert is firing
- Sent once if the OAM application is restarted and the alert is firing.

Examples

The following example sends all alerts to community “test” with address 10.10.10.10.

```
scheduler(config)# alert snmp-v3-destination 10.10.10.10 user
test auth-proto SHA auth-password test engine-id 0x01020304 privacy-protocol
 AES privacy-password  test
```

apply patches

Applies patches that are staged in the `/data/orchestrator/patches/` directory of the master VM.

This command should only be used by the Cisco TAC and Engineering team to address specific problems and debug the application.

Syntax

```
apply patches
```

Command Parameters

Table 13: Parameter Description

Command Parameter	Description
Service Name or Prefix	The exact name of the service to apply the patch or the prefix of the services to apply.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

This command should only be used at the recommendation of Cisco TAC and Engineering teams.

binding db-connection

Adds additional binding db connections from the DRA to a DRA binding database.

Syntax

```
binding db-connection { ipv4 | ipv6 | imsiapn | msisdnapn| slf }
 address port
```

Command Parameters

Table 14: Parameter Description

Command Parameter	Description
ipv4	Connection definition for the IPv4 binding database.
ipv6	Connection definition for the IPv6 binding database.

Command Parameter	Description
imsiapn	Connection definition for the IMSI-APN binding database.
msisdnapn	Connection definition for the MSISDN-APN binding database.
slf	Connection definition for the SLF database.
address	Address of the binding DRA database. This is either an IP address or an FQDN.
port	Port of the binding DRA database.

Command Mode

CONFIG

VNFs

DRA

Command Usage

Use the binding db-connection command to instruct the application on how to connect to the remote binding database. In general, there should be configuration lines entered per binding database type in order to support high availability.

Examples

The following configuration defines two redundant connections per database.

```
binding db-connection ipv6 172.16.82.195 27017
!
binding db-connection ipv6 172.16.82.196 27017
!
binding db-connection ipv4 172.16.82.195 27017
!
binding db-connection ipv4 172.16.82.196 27017
!
binding db-connection imsiapn 172.16.82.195 27017
!
binding db-connection imsiapn 172.16.82.196 27017
!
binding db-connection msisdnapn 172.16.82.195 27017
!
binding db-connection msisdnapn 172.16.82.196 27017
!
binding db-connection slf 172.16.82.195 27017
!
binding db-connection slf 172.16.82.196 27017
!
```

binding db-connection-settings

Used to configure the mongo connection settings.

Syntax

```
binding db-connection-settings { drasession | imsiapn | ipv4 | ipv6 | msisdnapn | range |
slf }
 connect-timeout connections-per-host max-wait-time socket-timeout

no binding db-connection-settings <database>
```

Command Parameters

Table 15: Parameter Description

Command Parameter	Description
drasession	Connection definition for the DRA session database.
imsiapn	Connection definition for the IMSI-APN binding database.
ipv4	Connection definition for the IPv4 binding database.
ipv6	Connection definition for the IPv6 binding database.
msisdnapn	Connection definition for the MSISDN-APN binding database.
range	Port range to be used.
slf	Connection definition for the SLF database.
connect-timeout	Connection timeout in milliseconds. It is used only when establishing a new connection. Default: 500
connections-per-host	Maximum number of connections allowed per host for this MongoClient instance. Those connections are kept in a pool when idle. Once the pool is exhausted, any operation requiring a connection blocks waiting for an available connection. Default: 10
max-wait-time	Maximum wait time in milliseconds that a thread may wait for a connection to become available. Default: 500
socket-timeout	Socket timeout in milliseconds. It is used for I/O socket read and write operations. Default: 1000

Command Mode

CONFIG

VNFs

DRA

Command Usage

Use the binding db-connection-settings command to configure the mongo connection settings.

Examples

The following is an example:

```
admin@orchestrator(config)# binding db-connection-settings ?
Possible completions:
  drasession  imsiapn  ipv4  ipv6  msisdnapn  range  slf

admin@orchestrator(config)# binding db-connection-settings drasession ?
Possible completions:
  connect-timeout  connections-per-host  max-wait-time  socket-timeout  <cr>

admin@orchestrator(config-db-connection-settings- drasession)# connect-timeout ?
Possible completions:
  <int>[500]

admin@orchestrator(config-db-connection-settings- drasession)# connections-per-host ?
Possible completions:
  <int>[10]

admin@orchestrator(config-db-connection-settings- drasession)# max-wait-time ?
Possible completions:
  <int>[500]

admin@orchestrator(config-db-connection-settings- drasession )# socket-timeout ?
Possible completions:
  <int>[1000]
```

control-plane relay

Adds additional control-plane entries between two disconnected CPS vDRA sites.

Syntax

```
control-plane relay name address
address port port
```

Command Parameters

Table 16: Parameter Description

Command Parameter	Description
Name	A short name describing the connection.
address	An IP address or FQDN of the connection. IPv6 address must be enclosed in square brackets.

Command Parameter	Description
port (optional)	The destination port of the connection. Defaults to 6379 if not defined.

Command Mode

CONFIG

VNFs

DRA

Command Usage

Use the control-plane relay command to instruct the application how which links it should use to relay CPS vDRA control traffic. CPS vDRA control traffic is the traffic that describes the current endpoints within a site and the relay IPs for site to site communication. For a 2 site model there should be at least 4 entries defined in this definition (two for each site). For a 3 site model there should be at least 6 entries in this definition.

Examples

The following configuration adds a relay connection to siteA over address 10.10.10.10 port 6379.

```
scheduler(config)# control-plane relay siteA-1 address 10.10.10.10
port 6379
```

database cluster

Create a MongoDB database sharded cluster.

Syntax

```
database cluster name sharded-cluster-master
{true|false} no database cluster name
```

Command Parameters

Table 17: Parameter Description

Command Parameter	Description
Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records.

Command Parameter	Description
sharded-cluster-master	This parameter indicates if the current VNF will execute provisioning operations on the given cluster. If multiple VNF (s) have the same database cluster configuration only one of them should have the "sharded-cluster-master" set to true.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster command and sub-commands to instruct the application to provision a database cluster for use in application database operations.

Examples

The following is an example of creating a "binding" sharded cluster that is being managed by the current VNF.

```
scheduler(config)# database cluster binding
sharded-cluster-master true
```

database cluster *db-name* config-server *name*

Add a MongoDB configuration server process to the named database cluster.

Syntax

```
database cluster db-name config-server
name address address no database
cluster db-name config-server name
```

Command Parameters

Table 18: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name "binding" for storing binding and session records
Name	A short description of the config server name.

Command Parameter	Description
address	The IPv4 or IPv6 address of the config server. This parameter does not accept FQDN address format

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster config-server to add a config-server to the system.

Examples

The following is an example of adding a new config server to the “binding” cluster.

```
scheduler(config)# database cluster binding
config-server cfg-1 address 10.10.10.10
```

database cluster *db-name* config-server-seed *name*

Set the initial seed configuration server for boot-strapping the MongoDB replica set initialization process.

Syntax

```
database cluster db-name config-server-seed
name
```

Command Parameters

Table 19: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records
Name	A reference to the configuration server name that will act as the seed for bootstrapping the initial replica set.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster config-server-seed command to set the initial seed configuration server for boot-strapping the MongoDB replica set initialization process. This is required if a config server is set.

Examples

The following is an example of setting cfg-1 as the initial seed for a new config server to the “binding” cluster.

```
scheduler(config)# database cluster binding
config-server-seed cfg-1
```

database cluster *db-name* router *name*

Add a new MongoDB router to the named DB cluster.

Syntax

```
database cluster db-name
router name
```

Command Parameters

Table 20: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records
Name	A short description of the router name.
address	The IPv4 or IPv6 address of the config server. This parameter does not accept FQDN address format
port	The port to bind the router. Generally 27017

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster router command to add a router to named database cluster. Full initialization of database cluster requires at least one router to be defined and often for HA purposes multiple routers are required.

Examples

The following is an example of adding a router to the “binding” cluster.

```
scheduler(config)# database cluster binding
router router-1 address 10.10.10.10 port 27017
```

database cluster *db-name* shard *name*

Add a new MongoDB shard to the named DB cluster.

Syntax

```
database cluster db-name
shard name no database cluster
db-name shard name
```

Command Parameters

Table 21: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records
Name	A short description of the shard name.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster shard command to add a new shard to the named database cluster. Full initialization of database cluster requires at least the definition of one shard and often for scaling purposes multiple shards are required.

Examples

The following is an example of adding a shard to the "binding" cluster.

```
database cluster binding shard shard-1
```

database cluster *db-name* shard *shard-name* shard-server *name*

Add a new MongoDB shard to the named DB cluster.

Syntax

```
database cluster db-name shard
shard-name shard-server name
address address port port [arbiter
{true|false}] [memory_allocation_percent percent]
[priority priority] [voter {true|false}]
[storage-engine {IN_MEMORY|MMAPv1|WT}]
no database cluster db-name shard
shard-name server name
```

Note When creating replica set, ensure that all ports are the same, i.e, the replica set should have same port for ARBITER, PRIMARY, and SECONDARY.

Command Parameters

Table 22: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name "binding" for storing binding and session records
Shard Name	A short description of the shard name.
Name	A short description of the server name.
address	The IPv4 or IPv6 address of the router server. This parameter does not accept FQDN address format.
port	The port to bind the router. Generally -27017
arbiter	Indicates if this node is only an arbiter node.
memory_allocation_percent	Percent (expresses as a positive integer) of the amount of memory to allocate to the DB process for the in-memory storage option.
priority	Relative priority of the node in the shard

Command Parameter	Description
voter	Whether this node is a voter.
storage-engine	The storage engine to provision for the process. Valid values are: • IN_MEMORY - pure in memory storage • MMAPv1 – Memory mapped files • WT –wired tirger

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster shard server command to add a new server to named database cluster. Full initialization of database cluster requires at least the definition of one shard server and for HA at least 3 nodes are required.

Examples

The following is an example of adding a new shard to the “binding” cluster.

```
scheduler(config)# database cluster binding shard
shard-1 shard-server server-1 storage-engine WT address
10.10.10.10 port 27017
```

Note Ports to be used for all database operations must be in the range of 27017 to 27047. Ports outside the defined range are not supported since the application must limit the port mappings. The selected range is sufficient for 30 Mongo processes on a given node.

database cluster *db-name* shard *shard-name* shard-server-seed *name*

Set the initial seed shard server for boot-strapping the MongoDB replica set initialization process.

Syntax

```
database cluster db-name
shard shard-name shard-server-seed name
```

Command Parameters

Table 23: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application will use a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records
Shard Name	A short description of the shard name.
Name	A reference to the shard server name that will act as the seed for bootstrapping the initial replica set.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the database cluster shard-server-seed command to set the initial seed shard server for boot-strapping the MongoDB replica set initialization process. This is required if a shard is defined.

Note To create or add a member to an existing replica set, you must also run the Mongo console-based commands as shown: `mongo> rs.add("name")`

To remove a replica set or a shard in a sharded cluster case, remove the member from the Mongo console as shown: `mongo> rs.remove("name")`

You must also navigate to the container and the VM on which the member resides and clear the data manually. The data path is the same as the one that is used when the replica-set member is created. Typically, the path is `//mmapv1-tmpfs-2xxxx` where `2xxxx` is the port where the replica set member is started.

Examples

The following is an example of setting server-1 as the initial seed for a new shard called “shard-1” to the “binding” cluster.

```
scheduler(config)# database cluster binding
shard shard-1 shard-server-seed server-1
```

database cluster *<db name>* ipv6-zone-sharding true/false

Enable the zone-based sharding for IPv6. When zone-based sharding is enabled on IPv6 database, hash-based sharding can still be configured on other databases.

Syntax

```
database cluster <db name>
ipv6-zone-sharding true/false
```

Command Parameters

Table 24: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application uses a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records.
ipv6-zone-sharding	Enables (true) or disables (false) zone-based sharding for IPv6 database. Default: False

Command Mode

CONFIG

VNFs

DRA Binding

Command Usage

Use database cluster binding ipv6-zone-sharding to enable/disable zone sharding on IPv6 database.

Examples

The following is an example of enabling zone-based sharding for the IPv6 database in the cluster `binding`:

```
database cluster binding ipv6-zone-sharding true
```

database cluster *<db name>* ipv6-zones-range *<zone-name>* zone-range *<range-name>* start *<pool starting address>* end *<pool ending address>*

Create zones for IPv6 shards based on IPv6 pools, so that the primary member of the replica set for an IPv6 address resides at the same physical location as the PGW assigning addresses from the IPv6 pool. This results in local writes (and reads) for the IPv6 binding database.

Note It is possible to create multiple ranges for each zone. Configure the IPv6 ranges in short format only.

Syntax

```
database cluster <db name> ipv6-zones-range
<zone-name> zone-range <range-name>
start <pool starting address> end
<pool ending address>
```

Command Parameters

Table 25: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application uses a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records.
Zone name	A short name describing Zone name. Unique name to identify the zone that the shard configuration uses to map to zone.
Range name	A short name describing the range within the zone.
Pool Starting Address	The starting IPv6 address for the particular range that can be from same physical location as PGW.
Pool Ending Address	The ending IPv6 address for the particular range that can be from same physical location as PGW.

Command Mode

CONFIG

VNFs

DRA Binding

Command Usage

This command creates a zone and also creates ranges for the zone.

Examples

The following is an example of creating a IPv6 zone with name `pune` for the cluster `binding` and a range of 2003:3051:0000:0001 to 2003:3051:0000:0500 for the zone:

```
database cluster binding ipv6-zones-range pune
zone-range range1 start 2003:3051:0000:0001 end
2003:3051:0000:0500
```

database cluster *<db name>* shard *<shard name>* zone-name *<zone-name>*

Add shards to a zone.

Syntax

```
database cluster <db name> shard
<shard name> zone-name <zone-name>
```

Command Parameters

Table 26: Parameter Description

Command Parameter	Description
DB Name	A short name describing the DB cluster. Each application uses a set of pre-defined names and this name should match one of the application names. For example, DRA uses the name “binding” for storing binding and session records.
Zone name	A short name describing Zone name.
Shard name	A short description of the shard name.

Command Mode

CONFIG

VNFs

DRA Binding

Command Usage

Use the command to add the shard to a zone.

Examples

The following is an example of mapping the IPv6 zone with name `pune` with the shard `shard-1` in the cluster `binding`:

```
database cluster binding shard shard-1 zone-name pune
```

db connect admin

Connects to an underlying admin database.

Syntax

No additional arguments.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the db connect admin command to connect to the underlying admin database. Once within this database, the user will have read / write access to the admin database via a mongodb CLI. The capabilities of the mongodb CLI are not described in this document.

db connect binding

Connects to an underlying binding database.

Syntax

```
db connect binding { ipv4 | ipv6 | imsi-apn | msisdn-apn
| slf }
```

Command Parameters

Table 27: Parameter Description

Command Parameter	Description
ipv4	Connect to the IPv4 binding database.
ipv6	Connect to the IPv6 binding database.

Command Parameter	Description
imsi-apn	Connect to the IMSI-APN binding database.
msisdn-apn	Connect to the MSISDN-APN binding database.

Command Mode

OPERATIONAL

VNFs

DRA

Command Usage

Use the db connect binding command to connect to the underlying binding database. Once within this database, the user will have read / write access to the binding database via the mongodb CLI. The capabilities of the mongodb CLI are not described in this document.

db connect session

Connects to an underlying admin database.

Syntax

No additional arguments.

Command Mode

OPERATIONAL

VNFs

DRA

Command Usage

Use the db connect session command to connect to the underlying session database. Once within this database, the user will have read / write access to the session database via a mongodb CLI. The capabilities of the mongodb CLI are not described in this document.

debug packet-capture gather

Gathers all running packet captures.

Syntax

```
debug packet-capture gather directory directory
```

Command Parameters

Table 28: Parameter Description

Command Parameter	Description
directory	The directory to store the resultant pcap files. This directory is available for downloading via the web file download interface at https://<master ip>/orchestrator/downloads/debug/<directory>.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `debug packet-capture gather` to gather all completed or currently running pcaps. This command is sent to all machines with active tcpdump commands and stops the given commands. After all commands are stopped, the command will gather the resultant pcap files and make them available at https://<master ip>/orchestrator/downloads/debug/<directory>.

debug packet-capture purge

Purges all existing pcap files.

Syntax

```
debug packet-capture purge
```

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `debug packet-capture purge` after all relevant packet captures have been downloaded from the application. The system does not automatically purge packet captures. You need to manage the amount of space used by the packet captures using this command.

debug packet-capture start

Starts a packet capture on a given IP address and port.

Syntax

```
debug packet-capture start ip-address ip-address
 port port timer-seconds timer-seconds
```

Command Parameters

Table 29: Parameter Description

Command Parameter	Description
ip-address	The IP address to start the packet capture. This address can either be IPv4 or IPv6..
port	The port to start the packet capture.
timer-seconds	Duration to run the packet capture - measured in seconds

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the debug packet-capture start command to start a tcp-dump on the given IP address and port within the CPS cluster. The packet capture will run for the given timer period and then shutdown automatically. The packet captures can be gathered using the debug packet-capture gather command.

debug tech

Gather logs and debug information to support troubleshooting.

Syntax

```
debug tech
```

Command Parameters

None

Command Mode

OPERATIONAL – Not available via NETCONF/RESTCONF

VNFs

All

Command Usage

Use this command to gather logs and debug information to support troubleshooting.

The results of the command are available at https://<master ip>/orchestrator/downloads/debug/tech.

Examples

```
scheduler# debug tech
```

docker connect

Connects to a docker service and launches a bash shell running on the system.

Syntax

```
docker connect container-id
```

Command Parameters

Table 30: Parameter Description

Command Parameter	Description
container-id	The docker container to open a bash shell. Use the **show docker service** command to find the list of valid container-ids.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `docker connect` to open a bash shell within a container. This command is primarily used for advanced debugging of the system. Once within a container, you can execute Linux commands and interact with the running container processes.

docker restart

Restarts a docker service that is currently running.

Syntax

```
docker restart container-id container-id
```

Command Parameters

Table 31: Parameter Description

Command Parameter	Description
container-id	The docker container to restart. Use the **show docker service** command to find the list of valid container-ids.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `docker restart` to restart a running docker service. This command is primarily useful to restore a non-responsive service at the request of Cisco TAC or Cisco Engineering.

external-aaa pam gid-mapping

Configures the gid mapping for various group roles.

Syntax

```
external-aaa pam gid-mapping <gid:int> <group name>
```

Command Parameters

Table 32: Parameter Description

Command Parameter	Description
gid:int	GID mapping value.
group name	Group name for which gid mapping is required.

Command Mode

CONFIG

VNFs

All

Command Usage

Use `external-aaa pam gid-mapping` to configure LDAP user gid mapping for various group roles such as, grafana-admin, policy-admin, policy-ro, and so on.

Based on the roles configured for the LDAP user gid, access permissions can be set accordingly.

Example

```
admin@orchestrator(config)# external-aaa pam gid-mapping 1000 policy-admin
admin@orchestrator(config-gid-mapping-1000/policy-admin)# commit
Commit complete.
```

You can display the status of configuration by running the following command:

```
admin@orchestrator# show running-config external-aaa | tab
```

Sample Output:

```
admin@orchestrator# show running-config external-aaa | tab
GID   GROUP
--------------------
1000  policy-admin
```

license feature

Registers a system license.

Syntax

```
license feature id encrypted-license encrypted-license
no license feature id
```

Command Parameters

Table 33: Parameter Description

Command Parameter	Description
id	ID of the license as provided by Cisco.
encrypted-license	The encrypted license as provided by Cisco.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the `license feature` to add and remove licenses from the running system.

logger set

Sets the various log levels for application logging.

Syntax

`logger set` *`logger-name`* `{ trace | debug | info | warn | error | off }`

Command Parameters

Table 34: Parameter Description

Command Parameter	Description
logger-name	Name of the logger to enable at the given log level.
trace	Enables trace logging and higher.
debug	Enables debug logging and higher.
info	Enables info logging and higher.
warn	Enables warn logging and higher.
error	Enables error logging.
off	Turns off all logging for the logger.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `logger set` to enable various levels of application logging. The logger names are provided by Cisco per application and are not defined here.

Examples

The following is an example:

```
logger set com.broadhop debug
```

logger clear

Clears a log level defined using the `logger set` command.

Syntax

```
logger clear logger-name
```

Command Parameters

Table 35: Parameter Description

Command Parameter	Description
logger-name	Name of the logger to enable at the given log level.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `logger clear` to reset the logging level for an application logger to the default level. The current set of logger levels can be found using the `show logger level` command.

monitor log application

Tails the cluster wide application log.

Syntax

```
monitor log application
```

Command Mode

OPERATIONAL

VNFs

DRA

Command Usage

Use the `monitor log application` to tail the `consolidated-qns.log` running on the cc-monitor docker services. If the cc-monitor docker services are not running, this command will fail.

Examples

The following is an example:

```
scheduler# monitor log application
binding-s3.weave.local  2017-03-06 00:07:07,256 [LicenseManagerProxy] INFO
consolidated.sessions - TPS_COUNT:                              SESSION_COUNT:
             LICENSE_COUNT: 100000000
binding-s4.weave.local  2017-03-06 00:07:15,577 [LicenseManagerProxy] INFO
consolidated.sessions - TPS_COUNT:                              SESSION_COUNT:
             LICENSE_COUNT: 100000000
diameter-endpoint-s1.weave.local  2017-03-06 00:07:21,041 [LicenseManagerProxy] INFO
consolidated.sessions - TPS_COUNT:                              SESSION_COUNT:
```

monitor log container

Tails a specific docker container using the `monitor log container` command.

Syntax

`monitor log container` *container-id*

Command Parameters

Table 36: Parameter Description

Command Parameter	Description
container-id	The container's log file to monitor. Use the `show docker service` command to list the valid container-ids.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the `monitor log container` command to tail the docker container log. This will provide the output for all non-application messages for the given container.

Examples

The following is an example:

```
scheduler# monitor log container svn
<<< Started new transaction, based on original revision 94
     * editing path : __tmp_run_stage ... done.

------- Committed revision 94 >>>
```

```
<<< Started new transaction, based on original revision 95
     * editing path : __tmp_run_backup ... done.
```

monitor log engine

Tails the cluster wide engine log using the `monitor log engine` command.

Syntax

```
monitor log engine
```

Command Mode

OPERATIONAL

VNFs

DRA

Command Usage

Use the `monitor log engine` to tail the `consolidated-engine.log` running on the cc-monitor docker services. If the cc-monitor docker services are not running this command will fail.

nacm rule-list

Specifies access restrictions for a user group.

Verify the users in the group before applying restrictions. To specify restrictions for any group, ensure that the admin user is not part of that group. By default, admin user is configured in a each group.

Syntax

```
nacm rule-list <rule-name>
group <group-name> cmdrule <cmdrule-name>
command <command to restrict>
access-operations exec action deny
```

Command Parameters

Table 37: Parameter Description

Command Parameter	Description
rule-list	Name of rule list.
group	Name of the group or list of groups to which the rules apply.
command	Command that is restricted for the user group.

Command Parameter	Description
access-operations	Used to match the operation that ConfD tries to perform. It must be one or more of the values from the accessoperations-type: create, read, update, delete, exec
action	If all of the previous fields match, the rule as a whole matches and the value of action (permit or deny) is taken. If a match is found, a decision is made whether to permit or deny the request in its entirety. If action is permit, the request is permitted; if action is deny, the request is denied.

Command Mode

CONFIG

VNFs

All

Command Usage

To delete the admin user from the read-only group, use the following command:

```
scheduler(config)#no nacm groups group crd-read-only
user-name admin
```

For the configuration to take effect, log out of the CLI session and log in again after configuring any nacm rule-list.

Examples

Restrict crd-read-only group from config command:

```
scheduler(config)#nacm rule-list crdreadgrp group
crd-read-only cmdrule denyconfig command config access-operations
exec action deny
scheduler(config-cmdrule-denyconfig)# commit
```

Restrict crd-read-only and policy-ro group from config command:

```
scheduler(config)#nacm rule-list readonly-restrict
group [ crd-read-only policy-ro ] cmdrule cfg-restrict command
config access-operations exec action deny
scheduler(config-cmdrule-cfg-restrict)#commit
```

Restrict crd-read-only and policy-ro group from docker command:

```
scheduler(config)#nacm rule-list readonly-restrict
group [ crd-read-only policy-ro ] cmdrule docker-restrict command
docker access-operations exec action deny
scheduler(config-cmdrule-docker-restrict)# commit
```

Restrict crd-read-only and policy-ro group from system stop command:

```
scheduler(config)#nacm rule-list readonly-restrict group
[ crd-read-only policy-ro ] cmdrule sys-stop command
"system stop" access-operations exec action deny
scheduler(config-cmdrule-sys-stop)# commit
```

Restrict crd-read-only and policy-ro group from system start command:

```
scheduler(config)#nacm rule-list readonly-restrict
group [ crd-read-only policy-ro ] cmdrule sys-start command
"system start" access-operations exec action deny
scheduler(config-cmdrule-sys-start)# commit
```

Restrict load override command for all the users including admin:

```
scheduler(config)#nacm rule-list readonly-restrict
group [ * ] cmdrule load-override command "load override"
access-operations exec action deny
scheduler(config-cmdrule-load-override)# commit
```

network dns server

Adds a network DNS server for the cluster to use.

Syntax

```
network dns server address no network
dns server address
```

Command Parameters

Table 38: Parameter Description

Command Parameter	Description
address	The IP address of the DNS server that the cluster can use. **Note** This address must be available to all servers within the cluster and is generally on an OAM network or the internal network.

Command Mode

CONFIG

VNFs

All

Command Usage

The network DNS server command triggers the addition of a DNS server to the DNS resolution that the application utilizes. These servers are added in the order they appear in the configuration to the DNS resolution.

Examples

The following example adds a DNS server:

```
scheduler(config)# network dns server 10.10.10.10
```

network dns host

Adds a network host to IP address mapping for the cluster to use.

Syntax

```
network dns host host domain address
address no network dns host host domain
```

Command Parameters

Table 39: Parameter Description

Command Parameter	Description
host	The host name of the host mapping to store.
domain	The domain name of the host mapping to store. Use local for hosts that do not have a domain name.
address	The IP address of the host / domain name mapping.

Command Mode

CONFIG

VNFs

All

Command Usage

The network DNS host command triggers the addition of a host / domain mapping to a specific IP address. This is useful when the upstream DNS services do not have a host / domain name mapping or upstream DNS server is not available to the cluster.

Examples

The following example adds a DNS server:

```
scheduler(config)# network dns host test local address 10.10.10.10
```

network virtual-service

Used to configure virtual floating IP address on various interfaces.

Syntax

```
network virtual-service name of floating ip floating-ip floating ip address mask
net mask digits broadcast broadcast address interface interface-id virtual-router-id
virtual router id tracking-service prefix of service to monitor for IP address
diameter-endpoint host ip address of host to put the floating ip priority priority of host

exit

host ip address of host to put the floating ip priority priority of host

commit

end
```

Command Parameters

Table 40: Parameter Description

Command Parameter	Description
name of floating ip	Name of the floating IP address. to be configured Virtual Network Service Name must contain a minimum of 1 character and a maximum length of 8 characters.
floating ip address	The floating IP address to manage with the virtual service.
net mask digits	The network mask (digits) for the floating IP address. Default: 24
broadcast address	The broadcast address of the floating IP.
interface-id	Interface ID.
virtual router id	virtual-router-id is the identity for a virtual router for hosts that are managed for VIP. Value range is from 0 to 255. For more details, refer to VRRP (Virtual Router Redundancy Protocol) RFC 3768 and keepalive documentation.
prefix of service to monitor for IP address	This parameter is a string used to define the service to be monitored.
ip address of host to put the floating ip	IP address of the host where floating IP is hosted.
priority of host	Priority of the host on which the service must run. Priority range is from 1 to 255. Higher the value, higher is the priority.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the `network virtual-service` command to configure virtual floating IP address on various interfaces that is managed using keepalive and the VRRP protocol. This command should be used in conjunction with the network virtual-service host command to assign floating IPs to given hosts.

Note To use within OpenStack, you must enable Protocol 112 on the security group – this is the VRRP protocol used by Keepalive. VRRP is configured as protocol number and not name. Hence, while configuring from dashboard, select protocol as 'Other' and in the text box below, enter 112 as protocol.

Examples

The following example creates a floating IP on two hosts:

Note Enter the command manually.

IPv4 VIP config:

```
scheduler(config)# network virtual-service GxVip12 floating-ip 172.22.33.51 mask 24 broadcast
 172.22.33.255 interface
ens161 virtual-router-id 1 tracking-service diameter-endpoint host 172.22.33.43 priority 2
exit
host 172.22.33.44 priority 1
commit
end
```

IPv6 VIP config:

```
scheduler(config)# network virtual-service RxVip12 floating-ip 2003:2235::51 mask 64 interface

ens192 virtual-router-id 2 tracking-service diameter-endpoint host 2003:2235::44 priority
2
exit
host 2003:2235::43 priority 1
commit
end
```

You can check the status of configuration on the scheduler by running the following command:

```
show running-config network
```

Sample Output:

```
network virtual-service GxVip12
 virtual-router-id 1
 floating-ip       172.22.33.51
 mask              24
 broadcast         172.22.33.255
 host 172.22.33.43
  priority 2
 !
```

```
 host 172.22.33.44
  priority 1
 !
!
```

Requirement

As a part of OpenStack configuration to have allowed-address-pairs configured on the VMs that are going to host the VIP.

Here is an example for ESC:

Under **vm_group** > **interfaces** > **interface**, you need to add the following configuration:

```
<allowed_address_pairs>
  <address>
    <ip_address>10.81.70.44</ip_address>
    <netmask>255.255.255.0</netmask>
  </address>
</allowed_address_pairs>
```

Note The above mentioned configuration needs to be done on all the interfaces of all the VMs where you want a virtual IP.

network virtual-service name host

Adds a new virtual-service floating IP address to the system.

Syntax

```
network virtual-service name host address
priority priority no network virtual-service
name host address
```

Command Parameters

Table 41: Parameter Description

Command Parameter	Description
name	The logical name of the virtual service floating IP. Virtual Network Service Name must contain a minimum of 1 character and a maximum length of 8 characters.
address	The IP of the host that should manage this floating IP.
priority	The priority of the host relative other hosts within the group. Default: 100

Command Mode

CONFIG

VNFs

All

Command Usage

Use this command to add new hosts to a virtual service. The hosts added will be start a Keepalive process to manage the floating IP via the VRRP process.

Examples

The following example adds a floating IP on a host:

```
scheduler(config)# network virtual-service
test host 10.84.100.136 priority 100
```

ntp server

Creates an NTP server for the system to synchronize system clocks.

Syntax

```
ntp server name address address
```

Command Parameters

Table 42: Parameter Description

Command Parameter	Description
name	Name of the server.
address	IP address or FQDN of the NTP server.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the `ntp server` command to synchronize the clocks of each virtual machine within the cluster. When this command is used, each node will run an NTP service. The NTP service is either a client or relay as described below:

- A relay node is a node that can reach at least one of the NTP servers defined in the configuration. The relay nodes are configured to point to the ntp servers defined in the server.

- A client node is an internal node that cannot reach an NTP server. The client nodes are configured to point to the relay nodes.

Examples

The following is an example:

```
scheduler(config)# ntp server server1 address 10.10.10.10
```

scheduling external-service

Creates a docker service that is external to the installed application.

Syntax

```
scheduling external-service name image image cap-add cap-add environment environment
host-network { true | false } port-mapping port-mapping run-level run-level scalable { true
 | false } scheduling-slot scheduling-slot volume volume
```

Command Parameters

Table 43: Parameter Description

Command Parameter	Description
name	Name of the service
image	Fully qualified image name.
scalable (optional)	Scale multiple instances across hosts. Default is false.
run-level (optional)	Relative run level between external services. Default is 0.
host-network (optional)	Bind to the host network. Default is to the overlay network.
volume (optional)	Volume mounts in the format is as follows: <host path>:<docker path>. Additional mounts are separated by ",".
port-mapping (optional)	Port mapping of the format is as follows: <external>:<internal>. Additional mounts are separated by ",".
cap-add (optional)	Linux capabilities to add to the container. Additional mounts are separated by ",".

Command Parameter	Description
scheduling-slot (optional)	Scheduling slot to start the container (for all containers). Use the **show running-config docker engine** command to view list of scheduling slots.
environment (optional)	Environment variables to export into the container in the format given below: <KEY>=<VALUE> Additional mounts are separated by ",".

Command Mode

CONFIG

VNFs

All

Command Usage

The `scheduling external-service` instructs the scheduling application to run the defined docker image on the given scheduling slots based on the configuration defined. Once scheduled the external-service appears in the `show scheduling status` and the `show docker service` commands.

scheduling vm-target

Calculates a vm-target for an external scaling system.

Syntax

```
scheduling vm-target name group-size group-size  k k  max  max min min  override override
 query query  scale-up-threshold scale-up-threshold
```

```
no scheduling vm-target name
```

Command Parameters

Table 44: Parameter Description

Command Parameter	Description
name	Name or identifier for the vm-target rule.
group-size (optional)	Size of the scaling group. Default is one
k (optional)	K value in an n + k redundancy model. Default is one.

Command Parameter	Description
max (optional)	Maximum value to calculate for the vm-target.
min (optional)	Minimum value to calculate for the vm-target.
override (optional)	Override value for the vm-target. This overrides anything the equation would calculate.
query	Query to calculate a raw scaling value.
scale-up-threshold	Divisor when calculating the scaling number. The query's raw value is divided by the scale-up-threshold to get a the value of n in an n+k redundancy model.

Command Mode

CONFIG

VNFs

All

Command Usage

The `scheduling vm-target` instructs the system to calculate VM scaling targets which can be used by the system to add and remove scaling VMs as required. The following algorithm is used to calculate the VM target for a given “name”:

vm-target(name) = roundup ((query value) / (scale-up-threshold))*group-size+K

show alert status

Displays the status of all alerts in the system. It displays either all alert statuses or alerts for a specific named alert.

Syntax

`show alert status` *`rule-name`*

Command Parameters

Table 45: Parameter Description

Command Parameter	Description
rule-name (optional)	Displays alert statuses for a given rule-name.

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show scheduling status
                              OUT
                 RUN          OF
MODULE INSTANCE LEVEL STATE DATE
-------------------------------------------------------
consul 1         50   RUNNING false
admin-db 1       75 RUNNING false
memcached-vip 1 100 RUNNING false
prometheus 1     100 RUNNING false
prometheus 2     100 RUNNING false
prometheus 3     100 RUNNING false
```

Table 46: Parameter Description

Parameter	Description
Name	Rule-name of the alert.
Event Host	Host where the alert was generated.
Status	Status of the alert. Valid values are: • firing • resolved
Message	Current alert message.
Update Time	Timestamp of the first alert message that transitioned to the given status.

show database status

Display the currently configured database clusters members.

Syntax

```
show database status
```

Command Parameters

Table 47: Parameter Description

Command Parameter	Description
Address	The address of the database process.

Command Parameter	Description
Port	The port the database service is running.
Name	Name of the database process.
Status	The current status of the mongo process. Valid states are: • CONNECTED – The mongo router is connected to the config servers • NOT_CONNECTED – The mongo router is not connected to the config servers • NO_CONNECTION – The process is not up or is not monitored • STARTUP – The DB node is in the STARTUP mode • PRIMARY – The DB node is the current PRIMARY • SECONDARY – The DB node is a SECONDARY node • RECOVERING – The DB node is currently RECOVERING from a restart or other failure • STARTUP2 – The DB node is in STARTUP2 mode • UNKNOWN – The DB node is in an UNKNOWN state • ARBITER – The DB node is currently an active ARBITER • NOT_INITIALIZED – The DB node is not initialized and pending initialization
Type	The type of the mongo process. Valid values are: • replica_set – a member of the replica set • config_server – a member of the config server replica set • mongos – a mongo router process
Cluster Name	The name of the cluster that owns the process.
Shard	The name of the associated shard.
Replica Set	The name of the replica set associated to the process.

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show database status
                                                                CLUSTER
ADDRESS       PORT   NAME    STATUS     TYPE          NAME  SHARD   REPLICA SET
-------------------------------------------------------------------------------------------
192.168.65.2  27018  shardA  PRIMARY    replica_set   test  shardA  rs-shardA
192.168.65.2  27019  -       PRIMARY    config_server test  cfg     test-configsrv
192.168.65.2  27017  -       CONNECTED  mongos        test  router-1 test-configsrv
```

show docker engine

Displays the status of the clusters docker engines.

Syntax

```
show docker engine
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show docker engine
                                  MISSED
ID                        STATUS  PINGS
-------------------------------------------
binding-73d3dc            CONNECTED  0
binding-8a8d17            CONNECTED  0
binding-c74547            CONNECTED  0
binding-dabba5            CONNECTED  0
control-0                 CONNECTED  0
control-1                 CONNECTED  0
control-2                 CONNECTED  0
diameter-endpoint-0       CONNECTED  0
diameter-endpoint-1       CONNECTED  0
diameter-endpoint-2       CONNECTED  0
diameter-endpoint-3       CONNECTED  0
master-0                  CONNECTED  0
session-shard-1-e079cf    CONNECTED  0
session-shard-2-80941f    CONNECTED  0
```

Table 48: Parameter Description

Parameter	Description
ID	The identifier within the cluster of the docker engine. Generally, this maps to the hostname where the engine resides.
Status	Indicates if the scheduling application is connected to the docker engine running on a host.
Missed Pings	The number of consecutive missed pings for a given host.

show docker service

Displays the currently running docker services.

Syntax

```
show docker service
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show docker service
MODULE   INSTANCE  NAME           VERSION              ENGINE         CONTAINER ID
STATE   MESSAGE PENALTY BOX
-------------------------------------------------------------------------------------------------------
admin-db    1     mongo-admin-a      3.4.0.0          control-0       mongo-admin-a
HEALTHY  false     -
admin-db    1     mongo-admin-arb    3.4.0.0          master-0        mongo-admin-arb
HEALTHY  false     -
admin-db    1     mongo-admin-b      3.4.0.0          control-1       mongo-admin-b
HEALTHY  false     -
admin-db    1     mongo-admin-setup  12.9.9-2017      master-0        mongo-admin-setup
HEALTHY  false     -
                                -03-03.123.797af71
binding     1     binding            12.9.9-dra.2017  binding-73d3dc  binding-s1
HEALTHY  false     -
                                -03-03.115.0f485ef
binding     1     session-router     3.4.0.0          binding-73d3dc  session-router-s1
HEALTHY  false     -
binding     2     binding            12.9.9-dra.2017  binding-8a8d17  binding-s2
HEALTHY  false
                                -03-03.115.0f485ef
```

Table 49: Parameter Description

Parameter	Description
Module	Scheduling module that is executing the docker service.
Instance	For scalable modules, the instance number that the service relates.
Name	Logical name of the service.
Version	Version of the image executing.
Engine	Engine identifier that is executing the docker service.
Container ID	Container id of the docker service.
State	Current state of the docker service.
Penalty Box	Indicates if the service is waiting to be rescheduled if an error occurred.
Message	Message related to the penalty box designation.

show history

Displays the history of commands executed on the system.

Syntax

```
show history
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show history
03-04 16:56:03 -- show docker service | include diameter
03-04 16:56:22 -- show docker service | include diameter | include diameter-endpoint-0
03-04 16:57:31 -- docker connect docker-host-info-s8
03-04 16:59:19 -- docker connect socket-forwarder-s1
03-04 17:01:02 -- ifconfig
03-04 17:01:22 -- docker connect socket-forwarder-s1
03-04 17:01:54 -- docker connect diameter-endpoint-s2
03-04 17:03:32 -- docker connect diameter-endpoint-s2
03-04 17:05:25 -- docker connect diameter-endpoint-s1
```

show license details

Displays the current license details installed on the system.

Syntax

```
show license details
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show license details
ID       DEFAULT  COUNT      EXPIRATION
-----------------------------------------------------
SP_CORE  true     100000000  2017-06-02T02:04:07+00:00
```

Table 50: Parameter Description

Parameter	Description
ID	ID of the license entry.
Default	Indicates if this is the default 90 day license installed on system install.
Count	Count for the given license.
Expiration	Expiration timestamp for the license.

show log application

Displays the application log in a viewer that enables you to scroll and search.

Syntax

```
show log application
```

Command Mode

OPERATIONAL

VNFs

DRA

show log engine

Displays the engine log in a viewer that enables you to scroll and search.

Syntax

```
show log engine
```

Command Mode

OPERATIONAL

VNFs

DRA

show logger level

Displays the current logger levels in the system that overrides the default logging.

Syntax

```
show logger level
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show logger level
Logger    Current Level
--------  ---------------
dra       warn
```

Table 51: Parameter Description

Parameter	Description
Logger	The logger that is overridden.
Current Level	The current level of logging.

show patches

Lists the patches that are in `/data/orchestrator/patches` directory.

Syntax

```
show patches
```

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The `show patches` indicates the patch that is loaded in the given patch directory and not a patch that is applied to the system .

show running-config binding db-connection-settings

Displays the binding DB connection settings.

Syntax

```
show running-config binding db-connection-settings
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show running-config binding db-connection-settings | tab
                           MAX
BINDING  CONNECT  SOCKET   WAIT  CONNECTIONS
TYPE     TIMEOUT  TIMEOUT  TIME  PER HOST
-------------------------------------------
drasession       500      1000     500   10
```

show scheduling effective-scheduler

Displays the effective scheduler running in the system.

Valid results are HA and AIO.

Syntax

```
show scheduling effective-scheduler
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show scheduling effective-scheduler
scheduling effective-scheduler HA
```

show scheduling status

Displays the currently loaded modules.

Syntax

```
show scheduling status
```

Command Mode

OPERATIONAL

VNFs

All

Examples

The following is an example:

```
scheduler# show scheduling status
                                                OUT
                                  RUN           OF
MODULE                  INSTANCE  LEVEL  STATE  DATE
-------------------------------------------------------
consul                  1         50     RUNNING  false
admin-db                1         75     RUNNING  false
memcached-vip           1         100    RUNNING  false
prometheus              1         100    RUNNING  false
```

```
prometheus            2         100    RUNNING  false
prometheus            3         100    RUNNING  false
```

Table 52: Parameter Description

Parameter	Description
Module	Module name that is running.
Instance	The instance number scheduled for scalable modules.
Run Level	The relative run level of the module compared to other modules. In an upgrade, the system reschedules from highest run level to lowest run level and in a downgrade the system schedules from low to high.
State	The current state of the module. Valid states are: • RUNNING • SCHEDULING • STOPPING
Out of Date	Indicates whether the software is out of date with the running system.

show scheduling vm-target

Displays the results of the scheduling vm-target calculation.

Syntax

```
show scheduling vm-target
```

Command Mode

OPERATIONAL

VNFs

All

Parameter Description

Parameter	Description
group	The vm-target group name that the count applies.
Count	The calculated count of VMs for scaling.

show system diagnostics

Shows the current diagnostics.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 53: Parameter Description

Command Parameter	Description
Node ID	ID of the node where the diagnostics was run.
Check	The ID of the check that was run.
IDX	For Checks that return multiple results the corresponding index number
Status	Indicates if the check is passing or not.
Message	The corresponding message for the diagnostic.

Examples

```
scheduler# show system diagnostics | tab
NODE        CHECK ID                  IDX  STATUS   MESSAGE


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

binding-s1  serfHealth                1    passing  Agent alive and reachable


binding-s1  service:cisco-policy-api  1    passing  TCP connect localhost:8080: Success


binding-s1  service:cisco-policy-app  1    passing  CLEARED: Session creation is allowed


binding-s1  service:cisco-policy-app  2    passing  CLEARED: -Dcom.broadhop.developer.mode
 is disabled
```

show system history

Shows the history of system events.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 54: Parameter Description

Command Parameter	Description
IDX	The index of the event in the system history log.
Event Time	Timestamp of the event in the system history log.
Module	The internal module that generated the history log entry.
Message	The message associated with the log entry.

Examples

```
scheduler# show system history
IDX  EVENT TIME                     MODULE         MESSAGE

---------------------------------------------------------------------------------------------------------
1    2017-02-04T02:04:02.469+00:00  system         System started

2   2017-02-04T02:04:29.021+00:00  docker-engine  Adding docker engine session-shard-2-80941f

3    2017-02-04T02:04:29.096+00:00  docker-engine  Adding docker engine diameter-endpoint-3

4    2017-02-04T02:04:29.187+00:00  docker-engine  Adding docker engine diameter-endpoint-2

5    2017-02-04T02:04:29.303+00:00  docker-engine  Adding docker engine binding-c74547

6    2017-02-04T02:04:29.375+00:00  docker-engine  Adding docker engine control-2

7   2017-02-04T02:04:29.503+00:00  docker-engine  Adding docker engine session-shard-1-e079cf

8    2017-02-04T02:04:29.583+00:00  docker-engine  Adding docker engine control-1

9    2017-02-04T02:04:29.671+00:00  docker-engine  Adding docker engine control-0

10   2017-02-04T02:04:29.751+00:00  docker-engine  Adding docker engine binding-dabba5
```

```
11   2017-02-04T02:04:29.843+00:00  docker-engine  Adding docker engine binding-73d3dc

12   2017-02-04T02:04:29.981+00:00  docker-engine  Adding docker engine binding-8a8d17
```

show system secrets open

Shows if the system secrets are unsealed.

This command returns true if the secrets are unsealed and false if they are still sealed. To open the system secrets, see system secrets unseal , on page 136.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system secrets open
system secrets open true
```

show system secrets paths

Shows the current set secrets.

This command does not show the value of the secrets only the path and if the value is readable by the system.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 55: Parameter Description

Command Parameter	Description
Path	The identifying path of the secret.
Status	Indicates if the path can be read by the system.

Examples

```
scheduler# show system secrets paths
PATH  STATUS
--------------
test  valid
```

show system software available-versions

Shows the list of available software versions to upgrade or downgrade a system.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system software available-versions
VERSION
-----------------------------------
12.9.9-dra.2017-03-03.115.0f485ef
```

show system software docker-repository

Shows the currently configured docker-repository.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system software docker-repository
system software docker-repository registry:5000
```

show system software version

Shows the currently installed software version.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system software version
system software version 12.9.9-dra.2017-03-03.115.0f485ef
```

show system software iso stage file

Displays the currently staged files in the /data/isos/staged-isos folder.

Syntax

```
show system software iso stage file
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Examples

The following example also shows a sample output:

```
scheduler# show system software iso stage file
NAME                  CREATED                    SIZE MB  MD5 SUM

-------------------------------------------------------------------------------------------
cisco-policy-dra.iso  2017-05-17T12:35:58+00:00  1100.04  c636794475b76e84041901b0ca3dcac4
```

Where:

- Name: The filename of the iso.
- Created: The date the file was created on the file system.
- Size MB: The size of the file in megabytes.
- MD5 Sum: The MD5 sum of the file.

show system software iso details

Displays the currently active ISOs that are loaded on the system.

Syntax

```
show system software iso details
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Examples

The following example also shows a sample output:

```
CATEGORY  NAME     VERSION  QUALIFIER    CREATED  ACTIVE  MB
-------------------------------------------------------------------------
product cisco-policy-dra 12.9.9   dra.2017-05-   2017-05  true   1102.9
                                    17.441.69    -17T13:
                                    68d89        4:15.708
                                                 +00:00
```

Where:

- Category: The type of ISO. Either product or extras. Extras can be used to load external docker images for use by external services.
- Name: The product name of the ISO

- Version: The version of the ISO
- Qualifier: The qualifier of the ISO
- Created Date: The creation date of the ISO on the file system
- Active: Indicates if the registry is currently pointing to the ISO to download images.
- Size: The size of the ISO on the file system.

show system status debug

Shows if the system is currently configured with debug tools.

Syntax

```
show system status debug
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Examples

The following example also shows a sample output:

```
scheduler# show system status debug
system status debug false
```

Where:

- Debug: Indicates if the system is configured to deploy containers with debug tools

show system status downgrade

Shows if the system is currently downgrading the installed software.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system status downgrade
system status downgrade false
```

show system status running

Shows if the system is currently running.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system status running
system status running true
```

show system status upgrade

Shows if the system is currently upgrading an installed software.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Examples

```
scheduler# show system status upgrade
system status upgrade false
```

statistics bulk file

Defines a new bulk statistics file that the system generates on a regular basis.

Syntax

```
statistics bulk file name header
 header query query format
format no bulk file name
```

Command Parameters

Table 56: Parameter Description

Command Parameter	Description
name	The base name of the bulk statistics file to create. The final file name generated has the following format:<name>-<timestamp in seconds>.csv
header	The exact text of the header to put at the start of all new files.
query	The Prometheus query to execute to build the bulk statistics. The query format is described in the Prometheus documentation:https://prometheus.io/docs/querying/basics/
format	The format of the output line. Each time series returned from the query that is executed will pass through the formatting string. Substitution variables appear as ${variable}. The following pre-defined variables exist in addition to the ones returned from Prometheus: • current-value – last value returned • max-value – max value over last 5 minutes • avg-value – average value over last 5 minutes • min-value – minimum value over last 5 minutes • timestamp – timestamp of when the sample was taken in the following format: yyyy-MM-dd'T'HH:mm:ss'Z'

Command Mode

CONFIG

VNFs

All

Command Usage

Use the bulk file command to define a bulk statistics file that supplements the default bulk statistics files created by the system. The format and queries are user defined.

Examples

The following example creates a bulk file on peer message rates:

```
statistics bulk file peer_tps
 query  "peer_message_total{remote_peer!=\"\"}"
 format ${app_id},${direction},${instance},${local_peer},
${remote_peer},${type},${current-value}
!
```

statistics bulk interval

Modifies the timer that the system uses to generate the bulk statistics that are defined via the bulk file command.

Syntax

```
statistics bulk interval interval no bulk interval
```

Command Parameters

Table 57: Parameter Description

Command Parameter	Description
interval	Timer length (in seconds) used to trigger a new bulk statistics file.

Command Mode

CONFIG

VNFs

All

Command Usage

Use the bulk interval command to control the timer length in triggering a new bulk statistics file.

Notes:

1. The generation of bulk statistics runs +/- 10 seconds of the interval.
2. The generation of bulk statistics is not synchronized to the minute.
3. The default interval, if not defined, is 300 seconds.

Examples

The following example creates a bulk file every 10 minutes:

```
scheduler(config)# bulk interval 600
```

statistics icmp-ping

Creates a probe that tests whether a host is up using ICMP ping.

Syntax

```
statistics icmp-ping address no statistics icmp-ping address
```

Command Parameters

Table 58: Parameter Description

Command Parameter	Description
address	The address to ping via ICMP. The resultant statistics are stored in the following metric: • probe_success • probe_duration_seconds • probe_ip_protocol

Command Mode

CONFIG

VNFs

All

Command Usage

Use the statistic icmp-ping command to instruct the monitoring system to ping the given address using the ICMP protocol. The IP address must be reachable via the master, control-a, and control-b hosts.

Examples

The following example creates an ICMP ping test:

```
scheduler(config)# statistics icmp-ping 10.10.10.10
```

statistics detail

Adds a statistics detail for the system to capture.

Syntax

```
statistics detail query category name query query format format scale scale
```

Command Parameters

Table 59: Parameter Description

Command Parameter	Description
category	Category of the statistic.
name	Name of the statistic.
query	Prometheus query to execute in order to retrieve the statistics.
format (optional)	Formatting rule for the statistic. The labels from the Prometheus query are substituted using the ${label} format.
scale (optional)	Scaling factor to take the raw value and scale to by the scale factor. A negative value divides by the scale factor and a positive value multiples by the scale factor.

Command Mode

CONFIG

VNFs

All

Command Usage

The statistics detail command triggers the application to monitor a given statistic and record it in memory and for reporting using the show statistics detail command. The values are refreshed every 10 seconds.

Examples

```
statistics detail query diameter success-message-tps
 query  "sum(rate(diameter_endpoint_request_total{result_code=\"2001\"}[10s])) by
(app_id,message_type)"
 format "${app_id} ${message_type}"
!
```

statistics icmp-ping

Creates a probe that tests whether a host is up using ICMP ping.

Syntax

```
statistics icmp-ping address no statistics icmp-ping address
```

Command Parameters

Table 60: Parameter Description

Command Parameter	Description
address	The address to ping via ICMP. The resultant statistics are stored in the following metric: • probe_success • probe_duration_seconds • probe_ip_protocol

Command Mode

CONFIG

VNFs

All

Command Usage

Use the statistic icmp-ping command to instruct the monitoring system to ping the given address using the ICMP protocol. The IP address must be reachable via the master, control-a, and control-b hosts.

Examples

The following example creates an ICMP ping test:

```
scheduler(config)# statistics icmp-ping 10.10.10.10
```

statistics summary

Adds a statistics summary for the system to capture.

Syntax

```
statistics summary query category name query query scale scale
```

Command Parameters

Table 61: Parameter Description

Command Parameter	Description
category	Category of the statistic.
name	Name of the statistic.

Command Parameter	Description
query	Prometheus query to execute in order to retrieve the statistics.
scale (optional)	Scaling factor to take the raw value and scale to by the scale factor. A negative value divides by the scale factor and a positive value multiples by the scale factor.

Command Mode

CONFIG

VNFs

All

Command Usage

The statistics summary command triggers the application to monitor a given statistic and record it in memory and for reporting using the show statistics summary command. The values are refreshed every 10 seconds.

The summary command does not support "group by" operations to show multiple lines from a single query.

Examples

```
statistics summary query diameter tps
 query "sum(rate(diameter_endpoint_request_total{result_code=\"2001\"}[10s]))"
!
```

system abort-downgrade

Stops a downgrade that is in progress.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The system abort-downgrade command stops the current rolling downgrade of the system. This command is only available when the system is in the process of downgrading and is not available after the downgrade is complete. Once this command is issued, system upgrade , on page 141 command should be issued to revert this software to the previous version.

system abort-upgrade

Stops an upgrade that is in progress.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Usage Guidelines

The system abort-upgrade command stops the current rolling upgrade of the system. This command is only available when the system is in the process of upgrading is not available after the upgrade is complete. Once the command is issued, system downgrade, on page 131 command should be issued to revert this software to the previous version.

system downgrade

Downgrades the system to a new software version.

Syntax

```
system downgrade version version
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 62: Parameter Description

Command Parameter	Description
Version	The new software version to install into the system.

Command Usage

The system downgrade command installs new software on the system using a rolling downgrade approach to minimize service interruption. Care must be taken to ensure that the system downgrade command is used when moving from a higher software version to a lower version of the software. The rolling downgrade

upgrades the software modules in startup order. After the command is issued, the CLI disconnects while the CLI software is restarted. The CLI generally becomes available within 30 seconds. Once the CLI becomes available, the status of the upgrade can be monitored using the show scheduling status, on page 115 command.

Examples

```
system downgrade version 12.9.9-dra.2017-03-03.115.0f485ef
```

system disable-debug

Disables debug tools in deployed containers.

Syntax

```
system disable-debug
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the system disable-debug command to turn off debugging tools on newly launched containers.

Examples

The following example disables debug tools:

```
scheduler# system disable-debug
```

system disable-external-services

Disables external services that are currently running in the system.

Syntax

```
system disable-external-services
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the system disable-external-services to stop all services registered with the scheduling external-service command.

Examples

The following example disables external services:

```
scheduler# system disable-external-services
```

system enable-debug

Enables debug tools in deployed containers.

Syntax

```
system enable-debug
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the system enable-debug command to turn on debugging tools on newly launched containers.

Examples

The following example enables debug tools:

```
scheduler# system enable-debug
```

system enable-external-services

Enable external registered services.

Syntax

```
system enable-external-services
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Command Usage

Use the system enable-external-services command to enable external services that are currently registered with the scheduling external-service command.

Examples

The following example enables external services:

```
scheduler# system enable-external-services
```

system secrets add-secret

Adds a secret to the system.

Syntax

```
system add-secret path path secret secret
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 63: Parameter Description

Command Parameter	Description
Path	The identifying path of the secret to add.
Secret	The clear text value of the secret to add.

Command Usage

The system add-secret command adds a secret to the system. This command is available only if the secrets are open. See show system secrets open , on page 119.

system secrets remove-secret

Removes a secret from the system.

Syntax

```
system remove-secret path path
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 64: Parameter Description

Command Parameter	Description
Path	The identifying path of the secret to remove.

Command Usage

The system remove-secret command removes a secret from the system. This command is available only if the secrets are open. See show system secrets open , on page 119.

system secrets set-passcode

Overwrites the current passcode that is used to encrypt or decrypt the master key for the secrets.

Syntax

```
system secrets set-passcode passcode
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 65: Parameter Description

Command Parameter	Description
Passcode	The new passcode to seal the secrets.

Command Usage

The system secrets command is used to change the passcode to unlock the secrets stored within the operational database. All secrets are encrypted using a randomly generated master-key that is encrypted/decrypted by the end-user provided passcode. If the passcode is lost, then the secrets currently stored are not recoverable. This command is available only if the secrets are open. See show system secrets open , on page 119.

system secrets unseal

Unseals the secrets if a non-default passcode is used to seal the secrets.

Syntax

```
system secrets unseal passcode passcode
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 66: Parameter Description

Command Parameter	Description
Passcode	The passcode to unseal the secrets.

Command Usage

The system secrets unseal command is used to unlock any stored secrets so that they can be shared with services that require a clear text secret or password. An example of this is a database connection password.

system software iso stage clean

Remove all downloaded ISOs from the stage directory.

Syntax

```
system software iso stage clean
```

Command Parameters

None

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The system software iso stage clean command removes all files that have been staged in the hosts /data/isos/staged-isos/ directory. This command should be run after an ISO file has been uploaded via the system software iso load command.

Examples

```
scheduler# system software iso stage clean
```

system software iso stage pull

Downloads a software ISO to the stage directory on the host.

Syntax

```
system software iso stage pull URL
```

Command Parameters

Table 67: Parameter Description

Command Parameter	Description
URL	The URL to download into the hosts /data/isos/staged-isos/ directory. If the URL ends with the zsync suffix, then the zsync command is invoked to retrieve the file.

Command Mode

OPERATIONAL - Not available via NETCONF/RESTCONF

VNFs

All

Command Usage

Invocation of the command downloads the given URL to the /data/isos/staged-isos/ directory. After invocation of this command, invocation of the show system software iso stage file command shows details of the downloaded file and the system software iso load command loads the file into the system.

Examples

The following example also shows a sample output:

```
scheduler# system software iso stage pull
http://171.70.34.121/microservices/latest/cisco-policy-dra.iso
--2017-05-17 15:08:39--  http://171.70.34.121/microservices
/latest/cisco-policy-dra.iso
Connecting to 171.70.34.121:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1153468416 (1.1G) [application/octet-stream]
Saving to: 'cisco-policy-dra.iso'

cisco-policy-dra.iso                                4%[=====>

                                ]  45.85M  4.07MB/s    eta 4m 27s
```

system software iso activate

Activate an existing ISO.

Syntax

```
system software iso activate category
[product|extras] name name version
version qualifier qualifier
```

Command Parameters

Table 68: Parameter Description

Command Parameter	Description
Category	The category to load the ISO. Either product or extras can be selected. The extras category represents a docker registry that contains external (non-product) docker images.
Name	The product name of the ISO to activate.
Version	The version of the ISO to activate
Qualifier	The qualifier of the ISO to activate

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The system software iso activate command triggers the system to restart the local docker registry to point to the given ISO. This command should be run before upgrading or downgrading the software.

Examples

The following example loads and activates a product ISO:

```
scheduler# system software iso activate category
product name cisco-policy-dra version 12.9.9 qualifier
dra.2017-05-17.441.6968d89
```

system software iso delete

Deletes an existing ISO.

Syntax

```
system software iso delete category
[product|extras] name name version
version qualifier qualifier
```

Command Parameters

Table 69: Parameter Description

Command Parameter	Description
Category	The category to load the ISO. Either product or extras can be selected. The extras category represents a docker registry that contains external (non-product) docker images.
Name	The product name of the ISO to delete.
Version	The version of the ISO to delete
Qualifier	The qualifier of the ISO to delete

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The system software iso delete command triggers the system to remove the ISO. This command can only be run on non-active ISOs.

Examples

The following example deletes an ISO:

```
scheduler# system software iso delete
category product name cisco-policy-dra version 12.9.9
qualifier dra.2017-05-17.441.6968d89
```

system software iso load

Load a new ISO into the system.

Syntax

```
system software iso load category
[product|extras] file filename activate [true|false]
```

Command Parameters

Table 70: Parameter Description

Command Parameter	Description
Category	The category to load the ISO. Either product or extras can be selected. The extras category represents a docker registry that contains external (non-product) docker images.
Filename	The filename of the ISO to load.
Activate	Indicates whether the system should switch the internal docker registry to point to the new ISO.

Command Mode

OPERATIONAL

Command Usage

The system software iso load command triggers unpacking of the staged ISO into a permanent location on the host. This command is executed before a system upgrade command can be executed.

Examples

The following example loads and activates an ISO:

```
scheduler# system software iso load category
product file cisco-policy-dra.iso activate true
```

system start

Starts all the services on a system that has been currently stopped.

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Usage Guidelines The system start command performs a controlled startup of the system by starting all the services in a rolling fashion taking into account various service dependencies.

system stop

Stops all the services on the system (excluding the CLI, NETCONF, and RESTCONF service).

Syntax

There are no arguments for this command.

Command Mode

OPERATIONAL

VNFs

All

Command Usage

The system stop commands performs a controlled shutdown of the system by stopping all the services in the reverse order of start-up.

Note For ephemeral databases (such as session), all data is lost on a system stop command.

system upgrade

Upgrades the system to a new software version.

Syntax

```
system upgrade version version
```

Command Mode

OPERATIONAL

VNFs

All

Command Parameters

Table 71: Parameter Description

Command Parameter	Description
Version	The new software version to install into the system.

Command Usage

The system upgrade command installs new software on the system using a rolling upgrade approach to minimize service interruption. Care must be taken to ensure that upgrade command is used when moving from a lower software version to a higher version of the software. The rolling upgrade upgrades the software modules in reverse start-up order. After the command is issued, the CLI disconnects while the CLI software is restarted. The CLI generally become available within 30 seconds. Once the CLI becomes available, the status of the upgrade can be monitored using the show scheduling status, on page 115 command.

Examples

```
system upgrade version 12.9.9-dra.2017-03-03.115.0f485ef
```