

Managing CPS vDRA Cluster

- Accessing CPS vDRA Management CLI, on page 1
- Starting CPS vDRA Cluster, on page 4
- Stopping Application Services In CPS vDRA Cluster, on page 5
- Starting Services In CPS vDRA Cluster, on page 5
- Stopping External Services In CPS vDRA Cluster, on page 5
- Starting External Services In CPS vDRA Cluster, on page 5
- Restarting An Individual Docker Service, on page 5
- CPS External Authentication and Authorization, on page 6
- vDRA Containers, on page 7
- Installing New Software Images, on page 13
- Upgrading to New Software Version, on page 13
- Downgrading to Previous Software Version, on page 14

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:

Procedure

Step 1 Enter the following URL in Firefox or Chrome:

https://<masterip>/

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

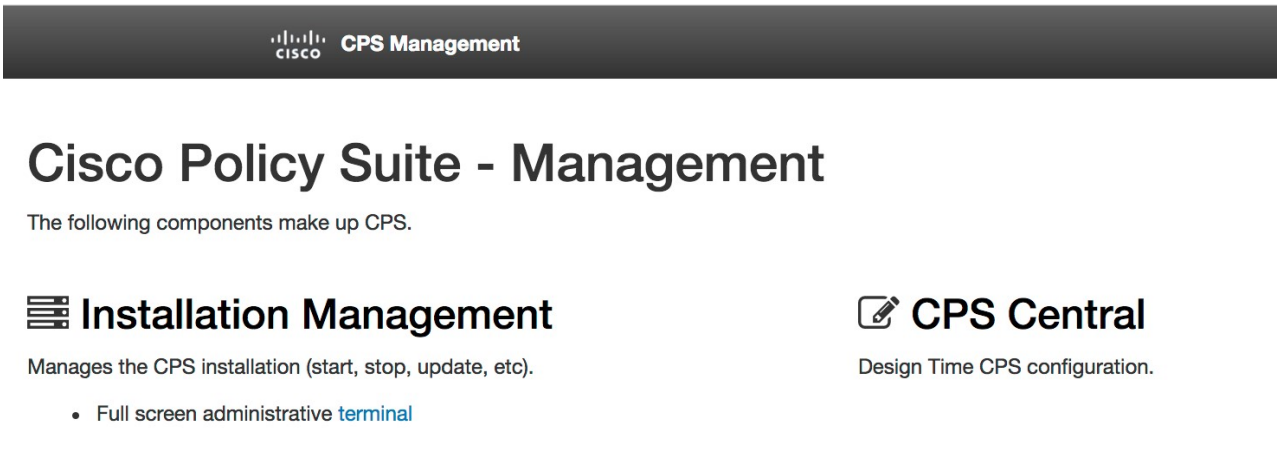
Step 3 Follow the Installation Management hyperlink in the following screen:

Figure 1: CPS DRA Login



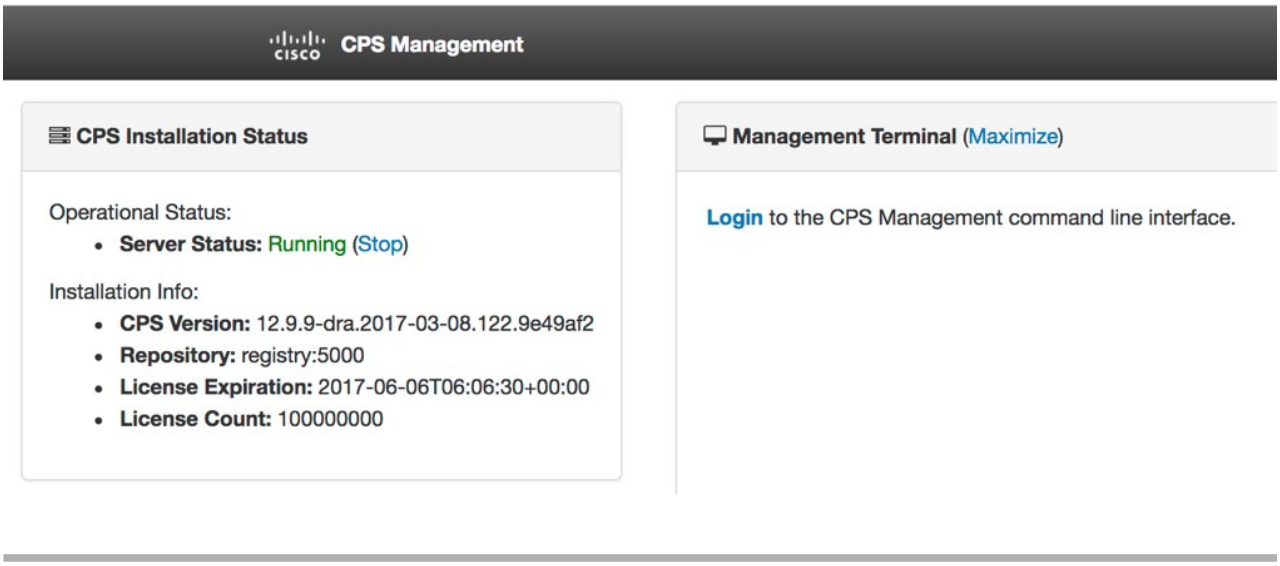
Step 4 In the Management screen, click the **Login** link to display the in-browser terminal window.

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.
    a. 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.
    b. 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.
    a. 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:

Procedure

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      3.6.9.0          aio     mongo-admin-a      HEALTHY
false    -
admin-db         1         mongo-admin-arb    3.6.9.0          aio     mongo-admin-arb    HEALTHY
false    -
admin-db         1         mongo-admin-b      3.6.9.0          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, on page 6.

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

Application/Service	CPS Role	Conf-D Groups
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

Module	Container	VM on which container runs	Description
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
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.

Module	Container	VM on which container runs	Description
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
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.

Module	Container	VM on which container runs	Description
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
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

Module	Container	VM on which container runs	Description
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 .
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.

Module	Container	VM on which container runs	Description
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
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

Module	Container	VM on which container runs	Description
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:

Procedure

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 New Software Version

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

Before you begin

Take a snapshot of the consul state to be used in case a rollback is required.

1. Login to CLI mode.

   ```
   docker connect consul-1
   ```

2. Take the backup and exit the CLI mode.

   Example:

   ```
   consul snapshot save <SITE-2-19.4-DBVNF-consul-backup.snap>
   ```

3. Copy the consul snapshot from orchestrator container to master VM.

   Example:

   ```
   docker cp consul-1:/ SITE-2-19.4-DBVNF-consul-backup.snap
   ```

4. Copy the backup to installer VM.

   Example:

```
scp -i cps.pem <backupdirectorypath>/SITE-2-19.4-DBVNF-consul-backup.snap
cps@<installerip>:/home/cps
```

Procedure

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, on page 14:

```
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:

Before you begin

Make sure older version consul snapshot is listed by executing `consul list-snapshots` command.

If the snapshot is not available, copy the older version consul snapshot taken Upgrading to New Software Version, on page 13 to the directory `/data/orchestrator/config/snapshot-consul` in master VM.

Trigger the DRA App VNF downgrade to older version (for example, 19.4.0 release) with consul downgrade (entire ISO downgrade) using `system downgrade version <version-qualifier> consul-downgrade true snapshot-name <snapshot-name>` command.

Example: `system downgrade version 19.4.0-20200625_121852.7720 consul-downgrade true snapshot-name SITE-2-19.4-DRAVNF-consul-backup.snap`

Procedure

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:

```
system abort-upgrade
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 2, on page 15:

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

At this point the application begins downloading the new scheduling and application images from the on-board Docker Registry. The download takes several seconds and the scheduler application disconnects and restarts. You must re-login after the disconnect occurs.



Note During downgrade, make sure consul is using the proper snapshot file after downgrade. If a consul snapshot was taken before the upgrade to the running version, find the list of available consul snapshots using the following command:

```
scheduler# consul list-snapshots
```

Select the correct consul snapshot for the version to be downgraded and downgrade DRA and consul using the following command:

```
scheduler# system downgrade version 12.9.9-dra.2017-03-08.122.9e49af2 consul-downgrade true
snapshot-name 12.9.9-dra.snap
```

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.