

Managing CPS Interfaces and APIs

- CPS Interfaces and APIs, on page 1
- Multi-user Policy Builder, on page 32
- Control Center Access, on page 34
- Enable Authentication for Unified API, on page 38
- Unified API Security: Access Privileges, on page 40
- Enabling Unified API Access on HTTP Port 8080, on page 42
- TACACS+, on page 44
- CRD APIs, on page 48
- Policy Builder Publish and CRD Import/Export Automation, on page 66
- Remove Traces of Old Policy Director (LB) VIPs, on page 68

CPS Interfaces and APIs

CPS includes southbound interfaces to various policy control enforcement functions (PCEFs) in the network, and northbound interfaces to OSS/BSS and subscriber applications, IMSs, and web applications.

Control Center GUI Interface

Purpose

Cisco Control Center enables you to do these tasks:

- Manage subscriber data, that is, find or create and edit information about your subscribers.
- View subscriber sessions.
- View system sessions.
- Populate custom reference data (CRD) tables.

URL and Port

HA: https://<*lbvip01*>:443

Protocol

HTTPS/HTTP

Accounts and Roles

There are two levels of administrative roles supported for Control Center: Full Privilege and View Only. The logins and passwords for these two roles are configurable in LDAP or in /etc/broadhop/authentication-password.xml.

- Full Privilege Admin Users: These users can view, edit, and delete information and can perform all tasks. Admin users have access to all screens in Control Center.
- View Only Admin Users: These users can view information in Control Center, but cannot edit or change information. View only administrators have access to a subset of screens in the interface.

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. The specific APIs are outlined in a later section in this guide.

URL and Port

HA: https:// <*lbvip01*>:443/custrefdata

A validation URL is:

HA: https:// <*lbvip01*>:8443/custrefdata

Protocol

HTTPS/HTTP

Accounts and Roles

Security and account management is accomplished by using the haproxy mechanism on the platform Policy Director (LB) by defining user lists, user groups, and specific users.

On Cluster Manager: `/etc/puppet/modules/qps/templates/etc/haproxy/haproxy.cfg`

Configure HAProxy

Update the HAProxy configuration to add authentication and authorization mechanism in the CRD API module.

1. Back up the `/etc/haproxy/haproxy.cfg` file.
2. Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 and add a userlist with at least one username and password as shown:

```
userlist <userlist name>
user <username1> password <encrypted password>
```

Use the following step to generate a encrypted password hash:

a. Execute `/var/qps/install/current/scripts/bin/support/generate_encrypted_password.sh` script to get encrypted password.

b. After script execution the encrypted password will be like below.

```
+-------------------------------------------------------------------------------------------------------+
| Fri May 29 11:43:47 UTC 2020
                                          |
| Encrypted key
                                                          |
|
$6$bc732ffd2a5ad85e$dYuQfGowAsAS6E2mQyWgGtcSUY4IKss11.4AY1u852gGwZzr4Y54rBdkHG6zQytFPXXDJGwknx.IYIeDeW.jP.

                          |
+-------------------------------------------------------------------------------------------------------+
```

3. Add the following line in frontend `https-api` to enable Authentication and Authorization for CRD REST API and create a new backend server as `crd_api_servers` to intercept CRD REST API requests:

```
mode http
acl crd_api path_beg -i /custrefdata/
use_backend crd_api_servers if crd_api
backend crd_api_servers
       mode http
       balance roundrobin
       option httpclose
       option abortonclose
       server qns01_A qns01:8080 check inter 30s
       server qns02_A qns02:8080 check inter 30s
```

4. Update frontend `https_all_servers` by replacing `api_servers` with `crd_api_servers` for CRD API as follows:

```
acl crd_api path_beg -i /custrefdata/

use_backend crd_api_servers if crd_api
```

5. Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 as follows:

a. Add at least one group with user in userlist created in *Step 2* as follows:

```
group qns-ro users readonly

group qns users apiuser
```

b. Add the following lines to the `backend crd_api_servers`:

```
acl authoriseUsers http_auth_group(<cps-user-list>) <user-group>

http-request auth realm CiscoApiAuth if !authoriseUsers
```

Map the group created in *Step 5* with the acl as follows:

```
acl authoriseUsers http_auth_group(<cps-user-list>) <user-group>
```

6. Add the following in the `backend crd_api_servers` to set read-only permission (GET HTTP operation) for group of users:

```
http-request deny if !METH_GET authoriseUsers
```

HAProxy Configuration Example:

```
userlist cps_user_list
       group qns-ro users readonly
       group qns users apiuser
     user readonly password
$6$xRtThhVpS0w4lOoS$pyEM6VYpVaUAxO0Pjb61Z5eZrmeAUUdCMF7D75B

XKbs4dhNCbXjgChVE0ckfLDp4T2CsUzzNkoqLRdn7RbAAU1
     user apiuser password
$6$xRtThhVpS0w4lOoS$pyEM6VYpVaUAxO0Pjb61Z5eZrmeAUUdCMF7D75B

XKbs4dhNCbXjgChVE0ckfLDp4T2CsUzzNkoqLRdn7RbAAU1
frontend https-api
   description API
   bind lbvip01:8443 ssl crt /etc/ssl/certs/quantum.pem

   mode http
   acl crd_api path_beg -i /custrefdata/
   use_backend crd_api_servers if crd_api

   default_backend api_servers
   reqadd X-Forwarded-Proto:\ https if { ssl_fc }

frontend https_all_servers
description Unified API,CC,PB,Grafana,CRD-API,PB-AP
bind lbvip01:443 ssl crt /etc/ssl/certs/quantum.pem no-sslv3 no-tlsv10
ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!
aNULL:!eNULL:!LOW:!     3DES:!MD5:!EXP:!PSK:!SRP:!DSS
mode http
acl crd_api path_beg -i  /custrefdata/
use_backend crd_api_servers if crd_api
backend crd_api_servers
       mode http
       balance roundrobin
       option httpclose
       option abortonclose
       server qns01_A qns01:8080 check inter 30s
       server qns02_A qns02:8080 check inter 30s
       acl authoriseReadonlyUsers http_auth_group(cps_user_list) qns-ro
       acl authoriseAdminUsers http_auth_group(cps_user_list) qns
       http-request auth realm CiscoApiAuth if !authoriseReadonlyUsers
!authoriseAdminUsers
       http-request deny if !METH_GET authoriseReadonlyUsers
```



Note The `haproxy.cfg` file is generated by the Puppet tool. Any manual changes to the file in lb01/lb02 would be reverted if the pupdate or vm-init scripts are run.

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

HA: https://<*lbvip01*>:9443/grafana

Protocol

HTTPS/HTTP

Accounts and Roles

An administrative user account must be used to add, modify, or delete Grafana dashboards or perform other administrative actions.

Refer to the *Graphite and Grafana* and *Prometheus and Grafana* chapters in this guide for details on adding or deleting these user accounts.

HAProxy

Purpose

Haproxy is a frontend IP traffic proxy process in lb01/lb02 that routes the IP traffic for other applications in CPS. The details of individual port that haproxy forwards is already described in other individual sections.

As per the Diameter configuration done, haproxy-diameter statistics will bind to one of the configurations and that URL will be displayed in `about.sh` output. For various options for Diameter configuration, refer to *Diameter Related Configuration* section in *CPS Installation Guide for VMware*.

More information about HAProxy is provided in the HAProxy.

Documentation for HAProxy is available at: http://www.haproxy.org/#docs

URL and Port

To view statistics, open a browser and navigate to the following URL:

- **For HAProxy Statistics**: http://<diameterconfig>:5540/haproxy?stats
- **For HAProxy Diameter Statistics**: http://<diameterconfig>:5540/haproxy-diam?stats

Accounts and Roles

Not applicable.

JMX Interface

Purpose

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

Resources to be managed / 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

External applications can be configured to monitor application over JMX. In addition to this, there are scripts provided by application that connects to application over JMX and provide required statistics/information.

Port

pcrfclient01/pcrfclient02:

- Control Center: 9045
- Policy Builder: 9046

lb01/lb02:

- iomanager: 9045
- Diameter Endpoints: 9046, 9047, 9048...

qns01/qns02/qns... : 9045

Ports should be blocked using firewall to prevent access from outside the CPS system.

Accounts and Roles

Not applicable.

Logstash

Purpose

Logstash is a process that consolidates the log events from CPS nodes into pcrfclient01/pcrfclient02 for logging and alarms. The logs are forwarded to CPS application to raise necessary alarms and the logs are stored at `/var/log/logstash/logstash.log`.

If logstash in not monitoring, then check the Policy Server (qns) process using `monit summary`.

```
monit summary
Monit 5.25.1 uptime: 19h 45m
```

Service Name	Status	Type
sav-pcrfclient01	OK	System

whisper	OK	Process
stale-session-cleaner-helper	Initializing	Process
stale-session-cleaner	Initializing	Process
snmpd	OK	Process
qns-2	OK	Process
qns-1	OK	Process
corosync	OK	Process
memcached	OK	Process
logstash	OK	Process
collectd	OK	Process
carbon-relay	OK	Process
carbon-cache-c	OK	Process
carbon-cache-b	OK	Process
carbon-cache	OK	Process
carbon-aggregator-b	OK	Process
carbon-aggregator	OK	Process
auditrpms.sh	OK	Process
aido_client	OK	Process
monitor-qns-2	OK	File
monitor-qns-1	OK	File
kpi_trap	OK	Program
db_trap	OK	Program
failover_trap	OK	Program
qps_process_trap	OK	Program
admin_login_trap	OK	Program
vm_trap	OK	Program
qps_message_trap	OK	Program
ldap_message_trap	OK	Program
logstash_process_status	OK	Program
monitor_replica	OK	Program
mon_db_for_lb_failover	OK	Program
mon_db_for_callmodel	OK	Program

cpu_load_monitor	OK	Program
cpu_load_trap	OK	Program
gen_low_mem_trap	OK	Program
auto_heal_server	OK	Program

Note On pcrfclient node, if Policy Server (qns) process is not running, 'logstash_process_status' program stops the logstash process so that the alarm is raised from another pcrfclient node.

CLI Access

There is no specific CLI interface for logstash.

Protocol

TCP and UDP

Ports

TCP: 5544, 5545, 7546, 6514

UDP: 6514

Accounts and Roles

Not applicable.

LDAP SSSD

Purpose

In CPS 14.0.0 and higher releases, SSSD based authentication is supported, allowing users to authenticate against an external LDAP server and gain access to the CPS CLI. SSSD RPMs and default `sssd.conf` file is installed on each CPS VM when you perform a new installation or upgrade CPS.

For more information, refer to the *CPS Installation Guide for VMware*.

`/etc/monit.d/sssd` file has been added with the following content so that SSSD is monitored by monit:

```
check process sssd with pidfile /var/run/sssd.pid
start program = "/etc/init.d/sssd start" with timeout 30 seconds
stop program = "/etc/init.d/sssd stop" with timeout 30 seconds
```

Also `/etc/logrotate.d/sssd` file has been added to rotate the SSSD log files. Here is the default configuration:

```
"
/var/log/sssd/*.log {
    daily
    missingok
    notifempty
    sharedscripts
```

```
    nodateext
    rotate 5
    size 100M
    compress
    delaycompress
    postrotate
        /bin/kill -HUP `cat /var/run/sssd.pid 2>/dev/null` 2> /dev/null || true
    endscript
}
"
```

Use the `monit summary` command to view the list of services managed by monit. Here is an example:

```
monit summary
The Monit daemon 5.17.1 uptime: 4d 2h 22m

Process 'whisper'                   Running
Process 'sssd'                      Running
Process 'snmptrapd'                 Running
Process 'snmpd'                     Running
Program 'vip_trap'                  Status ok
Program 'gr_site_status_trap'       Status ok
Process 'redis'                     Running
Process 'qns-4'                     Running
Process 'qns-3'                     Running
Process 'qns-2'                     Running
Process 'qns-1'                     Running
File 'monitor-qns-4'                Accessible
File 'monitor-qns-3'                Accessible
File 'monitor-qns-2'                Accessible
File 'monitor-qns-1'                Accessible
Process 'memcached'                 Running
Process 'irqbalance'                Running
Process 'haproxy-diameter'          Running
Process 'haproxy'                   Running
Process 'cutter'                    Running
Process 'corosync'                  Running
Program 'cpu_load_monitor'          Status ok
Program 'cpu_load_trap'             Status ok
Program 'gen_low_mem_trap'          Status ok
Process 'collectd'                  Running
Process 'auditrpms.sh'              Running
System 'lb01'                       Running
```



Important Setting of other configuration files to support LDAP based authentication and the changes required in `sssd.conf` file as per the customer deployment is out of scope of this document. For more information, consult your Cisco Technical Representative.



Restriction Grafana support LDAP authentication over httpd and does not use SSSD feature. Due to this, if LDAP server is down then grafana is not accessible for LDAP users.

CLI Access

No CLI is provided.

Port

Port number is not required.

Configure Policy Builder

Procedure

Step 1 To provide admin access, enter username in the following file:

`/var/www/svn/users-access-file`

Note
This action should be performed on pcrfclient and not on policy server (qns).

```
[groups]
admins = qns,qns-svn,sssd_pb_2
nonadmins = qns-ro
[/]
@admins = rw
@nonadmins = r
* = r
```

Step 2 Verify if you can export CRD data from the following link:

https://<server_ip>:443/central/

Configure Grafana

Procedure

Step 1 Bypass the first level authentication by updating the `/etc/httpd/conf.d/grafana-proxy.conf` file as follows:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
# Set root to <ip address>/grafana
ProxyPass /grafana http://127.0.0.1:3000
ProxyPassReverse /grafana http://127.0.0.1:3000
# Set authentication for Grafana
# 1) Use httpd authentication as a front-end to Grafana
# 2) Remove header since Grafana is configured for anonymous
# authentication and will fail with a pass-thru header
#
# Notice: scope of authentication and header is limited to Grafana
# to avoid conflicts with other applications. Apache configuration
# in this file is global unless contained in the directive below.
<Location "/grafana">
    LoadModule headers_module modules/mod_headers.so
    Header set Access-Control-Allow-Origin "*"
    Header set Access-Control-Allow-Methods "GET, OPTIONS"
```

```
    Header set Access-Control-Allow-Headers "origin, authorization, accept"
    Header set Access-Control-Allow-Credentials true
    # Do not pass credentials to Grafana's anonymous authorization
    RequestHeader unset Authorization
    Satisfy Any
    #AuthName "Authentication Required"
    #AuthUserFile "/var/broadhop/.htpasswd"
    #Require valid-user
    #Order allow,deny
    # This is used for local calls to the API during puppet bring up
    Allow from 127.0.0.1
    #Satisfy Any
</Location>
```

Step 2 Restart httpd by running the following command:

```
/usr/bin/systemctl restart httpd
```

If `port already in use error` is displayed, execute the following steps:

a) Run the following command to get process ID:

```
ps -eaf | grep httpd
```

b) Run the following command to kill the pid:

```
kill -9 <pid>
```

Step 3 Update `/etc/grafana/grafana.ini` file to point to LDAP authentication instead of Basic Auth as follows:

```
################################### Basic Auth ##########################
[auth.basic]
# For CPS, trusted API requests come here and need local authentication
;enabled = true
################################### Auth LDAP ##########################
[auth.ldap]
enabled = true
config_file = /etc/grafana/ldap.toml
```

Step 4 Modify `/etc/grafana/ldap.toml` file to provide LDAP details (for example, search base dn, bind dn, group search base dn, member_of attribute) as follows:

```
# Set to true to log user information returned from LDAP
verbose_logging = true
[[servers]]
# Ldap server host (specify multiple hosts space separated)
host = "ldap_l.cisco.com"
# Default port is 389 or 636 if use_ssl = true
port = 10648
# Set to true if ldap server supports TLS
use_ssl = true
# set to true if you want to skip ssl cert validation
ssl_skip_verify = true
# set to the path to your root CA certificate or leave unset to use system defaults
#root_ca_cert = "/etc/openldap/certs/ldap_local.cer"

# Search user bind dn
bind_dn = "uid=admin,ou=system"
# Search user bind password
bind_password = 'secret'

# User search filter, for example "(cn=%s)" or "(sAMAccountName=%s)" or "(uid=%s)"
search_filter = "(uid=%s)"

# An array of base dns to search through
```

```
search_base_dns = ["ou=users,dc=sprint,dc=com"]
#search_base_dns = ["ou=groups,dc=sprint,dc=com"]

# In POSIX LDAP schemas, without memberOf attribute a secondary query must be made for
groups.
# This is done by enabling group_search_filter below. You must also set member_of= "cn"
# in [servers.attributes] below.
# Users with nested/recursive group membership and an LDAP server that supports
LDAP_MATCHING_RULE_IN_CHAIN
# can set group_search_filter, group_search_filter_user_attribute, group_search_base_dns
and member_of
# below in such a way that the user's recursive group membership is considered.
#
# Nested Groups + Active Directory (AD) Example:
#
# AD groups store the Distinguished Names (DNs) of members, so your filter must
# recursively search your groups for the authenticating user's DN. For example:
#
# group_search_filter = "(member:1.2.840.113556.1.4.1941:=%s)"
# group_search_filter_user_attribute = "distinguishedName"
# group_search_base_dns = ["ou=groups,dc=grafana,dc=org"]
#
# [servers.attributes]
# ...
# member_of = "distinguishedName"

## Group search filter, to retrieve the groups of which the user is a member (only set if
memberOf attribute is not available)
#group_search_filter = "(cn=%s)"
#group_search_filter = "(&(objectClass=*)(cn=%s))"
## Group search filter user attribute defines what user attribute gets substituted for %s
in group_search_filter.
## Defaults to the value of username in [server.attributes]
## Valid options are any of your values in [servers.attributes]
## If you are using nested groups you probably want to set this and member_of in
## [servers.attributes] to "distinguishedName"
group_search_filter_user_attribute = "cn"
## An array of the base DNs to search through for groups. Typically uses ou=groups
group_search_base_dns = ["ou=groups,dc=sprint,dc=com"]
#group_search_base_dns = ["cn=Roles,ou=groups,dc=sprint,dc=com"]

# Specify names of the ldap attributes your ldap uses
[servers.attributes]
name = "cn"
surname = "sn"
username = "uid"
member_of = "cn"
email = "email"

# Map ldap groups to grafana org roles
[[servers.group_mappings]]
group_dn = "cn=Admin,ou=groups,dc=sprint,dc=com"
org_role = "Admin"
# The Grafana organization database id, optional, if left out the default org (id 1) will
be used
# org_id = 1

[[servers.group_mappings]]
group_dn = "cn=User,ou=groups,dc=sprint,dc=com"
org_role = "Editor"

#[[servers.group_mappings]]
# If you want to match all (or no ldap groups) then you can use wildcard
```

```
#group_dn = "*"
#org_role = "Viewer"
```

Step 5 Restart Grafana server by running the following command:

```
service grafana-server restart
```

Step 6 Log in to Grafana using LDAP user credentials.

Mongo Database

Purpose

MongoDB is used to manage session storage efficiently and address key requirements: Low latency reads/writes, high availability, multi-key access and so on.

CPS support different models of mongo database based on CPS deployment such as, HA or Geo-redundancy. The database list is specific to your deployment.

To rotate the MongoDB logs on the Session Manager VM, open the MongoDB file by executing the following command:

cat /etc/logrotate.d/mongodb

You will have output as similar to the following:

```
{
daily
rotate 5
copytruncate
create 640 root root
sharedscripts
postrotate
endscript
}
```

In the above script the MongoDB logs are rotated daily and it ensures that it keeps the latest 5 backups of these log files.

HA

The standard definition for supported replica-set defined in configuration file. This configuration file is self-explanatory which contains replica-set, set-name, hostname, port number, data file path and so on.

Location: `/etc/broadhop/mongoConfig.cfg`

Table 1: HA MongoDB

Database Name	Port Number	Primary DB Host	Secondary DB Host	Arbiter	Purpose
session_cache	27717	sessionmgr01	sessionmgr02	pcrfclient01	Session database
balance_mgmt	27718	sessionmgr01	sessionmgr02	pcrfclient01	Quota/Balance database
audit	27725	sessionmgr01	sessionmgr02	pcrfclient01	Reporting database

Database Name	Port Number	Primary DB Host	Secondary DB Host	Arbiter	Purpose
spr	27720	sessionmgr01	sessionmgr02	pcrfclient01	USuM database
cust_ref_data	27717	sessionmgr01	sessionmgr02	pcrfclient01	Custom Reference Data

Note The list provided in the Table 1: HA MongoDB, on page 13 is for reference purposes only.

Note The port number configuration is based on what is configured in each of the respective Policy Builder plug-ins. Refer to the *Plug-in Configuration* chapter of the *CPS Mobile Configuration Guide* for correct port number and ports defined in mongo configuration file.

CLI Access

Use the following commands to access the MongoDB CLI:

HA

Login to pcrfclient01 or pcrfclient02 and run: `diagnostics.sh --get_replica_status`

This command will output information about the databases configured in the CPS cluster.

Note If a member is shown in an unknown state, it is likely that the member is not accessible from one of other members, mostly an arbiter. In that case, you must go to that member and check its connectivity with other members.

Also, you can login to mongo on that member and check its actual status.

Protocol

Not applicable.

Port

Not applicable.

Accounts and Roles

Restrict MongoDB access for Readonly Users: If firewall is enabled on system, then on all VMs for all readonly users, IP table rule will be created for outgoing connections to reject outgoing traffic to MongoDB replica sets.

For example, rule similar to the following is created.

```
REJECT tcp -- anywhere sessionmgr01 tcp dpt:27718 owner GID match qns-ro reject-with
icmp-port-unreachable
```

With this, qns-ro user has restricted MongoDB access on sessionmgr01 on port 27718. Such rules are added for all readonly users who are part of qns-ro group for all replica sets.

Adding New Replica-set Members

Caution The following procedure must be performed only during a planned Maintenance Window (MW).

Note The procedure is for reference purposes only. Contact your Cisco Account representative before running the procedure.

Procedure

Step 1 Update the `mongoConfig.cfg` file with the new replica-set members to be configured.

Step 2 Login to the Cluster Manager VM of the site where you want to add new replica-set members.

Step 3 Take the backup of the current `/etc/broadhop/mongoConfig.cfg` file.

```
/bin/cp /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg.$(date +\%Y-\%m-\%d).backup
```

Step 4 Copy the updated `mongoConfig.cfg` file to `/etc/broadhop/`.

If the file is located on a remote machine, then `scp` *<user@vm:updated mongoConfig.cfg file_path>* `/etc/broadhop/`

If the file is present on the current VM but at a different location, then `/bin/cp` *<updated mongoConfig.cfg file_path>* `/etc/broadhop/`

Step 5 Verify that the Session Manager and arbiter VMs have sufficient space available to create the new replica-set member.

The verification command depends on where all the new replica-set members will get created. Here is an example in case all the Session Managers VMs are updated.

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "df -h";done
```

For arbiter VM, you have to login to each VM and use - `df -h` command to verify space availability.

Step 6 Verify if there are existing `/var/tmp/stopped-`*<PORT>* entries on Session Managers and arbiter VMs.

The verification command depends on where all the new replica-set members will get created. Here is an example in case all the Session Managers VMs are updated.

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/tmp/stopped-*"; done
```

For arbiter VM, you have to login to each VM and use `ls -ltrh /var/tmp/stopped-*` command to see if any file exists.

a) If there is an existing `/var/tmp/stopped-<PORT>` file entry, delete those entries.

Here is an sample command to delete the entries from Session Manager VM:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -rf /var/tmp/stopped-*"; done
```

For arbiter VM, you have to login to each VM and use `"rm -rf /var/tmp/stopped-*"` command to remove the file entries

Step 7 Verify whether `mongoConfig.cfg -*` files exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to verify whether `mongoConfig.cfg -*` files exists:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"ls -ltrh /var/aido/*"` command.

a) Delete all `mongoConfig.cfg -*` files that exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to delete `mongoConfig.cfg -*` files that exist:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -f /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"/bin/rm -f /var/aido/*"` command to remove `mongoConfig.cfg -*` files.

Step 8 Execute `copytoall.sh` to copy the updated `/etc/broadhop/mongoConfig.cfg` from Cluster Manager to all the VMs.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg
```

Step 9 SSH to remote Site2/Cluster2 and take the backup of `mongoConfig.cfg` file.

Step 10 Copy the `mongoConfig.cfg file` from local Site1/Cluster1 to remote Site2/Cluster2 on `/etc/broadhop`.

```
scp root@<local_site_cluman_ip>:/etc/broadhop/mongoConfig.cfg /etc/broadhop/
```

Step 11 Verify if there are existing `/var/tmp/stopped-<PORT>` entries on Session Managers and arbiter VMs.

The verification command depends on where all the new replica-set members will get created. Here is an example in case all the Session Managers VMs are updated.

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/tmp/stopped-*"; done
```

For arbiter VM, you have to login to each VM and use `ls -ltrh /var/tmp/stopped-*` command to see if any file exists.

a) If there is an existing `/var/tmp/stopped-<PORT>` file entry, delete those entries.

Here is an sample command to delete the entries from Session Manager VM:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -rf /var/tmp/stopped-*"; done
```

For arbiter VM, you have to login to each VM and use `"rm -rf /var/tmp/stopped-*"` command to remove the file entries

Step 12 Verify the `mongoConfig.cfg -*` files under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to verify whether `mongoConfig.cfg -*` files exists:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"ls -ltrh /var/aido/*"` command.

a) Delete all `mongoConfig.cfg -*` files that exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to delete `mongoConfig.cfg -*` files that exist:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -f /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use "/bin/rm -f /var/aido/*" command to remove mongoConfig.cfg -* files.

Step 13 Execute copytoall.sh to copy the updated /etc/broadhop/mongoConfig.cfg from Cluster Manager to all the VMs.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg
```

Step 14 On local Site1/Cluster1, execute build_etc.sh command to apply the new mongoConfig.cfg file to add the new replica-sets.

```
/var/qps/install/current/scripts/build/build_etc.sh
```

Step 15 On remote Site2/Cluster2, execute build_etc.sh command to apply the new mongoConfig.cfg file to add the new replica-sets.

```
/var/qps/install/current/scripts/build/build_etc.sh
```

Step 16 Wait for sometime (approx 5 minutes) and verify the new replica-set status by executing the following command.

```
diagnostics.sh --get_replica_status
```

Rollback Replica-set Members

Caution The following procedure must be performed only during a planned Maintenance Window (MW).

Note The procedure is for reference purposes only. Contact your Cisco Account representative before running the procedure.

Procedure

Step 1 Prepare a list of the replica-sets and setnames that you want to remove from the local and remote site.

Step 2 Execute the following command to remove the all the replica-sets identified in Step 1, on page 17.

```
build_set.sh --session --remove-replica-set --setname setxx --force
```

where, *setxx* with set name identified in Step 1, on page 17.

Step 3 Verify that the new replica-sets have been removed using diagnostics.sh --get_replica_status command.

Step 4 Copy the mongoConfig.cfg backup file saved in *Adding New Replica-set* earlier to /etc/broadhop on Cluster Manager VM of local site.

```
/bin/cp /etc/broadhop/mongoConfig.cfg.*.backup /etc/broadhop/
```

Step 5 Verify that the file mongoConfig.cfg has older configuration.

Note
You need to verify that the mongoConfig.cfg has older configuration manually.

Step 6 Execute `copytoall.sh` to copy the updated `/etc/broadhop/mongoConfig.cfg` from Cluster Manager to all the VMs.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg
```

Step 7 Verify whether `mongoConfig.cfg -*` files exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to verify whether `mongoConfig.cfg -*` files exists:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"ls -ltrh /var/aido/*"` command.

a) Delete all `mongoConfig.cfg -*` files that exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to delete `mongoConfig.cfg -*` files that exist:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -f /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"/bin/rm -f /var/aido/*"` command to remove `mongoConfig.cfg -*` files.

Step 8 SSH to remote site and rollback a `mongoConfig.cfg` from backup.

Step 9 Execute `copytoall.sh` to copy the updated `/etc/broadhop/mongoConfig.cfg` from Cluster Manager to all the VMs.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg
```

Step 10 Verify whether `mongoConfig.cfg -*` files exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to verify whether `mongoConfig.cfg -*` files exists:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "ls -ltrh /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"ls -ltrh /var/aido/*"` command.

a) Delete all `mongoConfig.cfg -*` files that exists under `/var/aido/` on each Session Manager and arbiter VM.

Here is an sample command to delete `mongoConfig.cfg -*` files that exist:

```
for i in $(hosts-sessionmgr.sh); do echo $i; ssh $i "rm -f /var/aido/*"; done
```

For arbiter VM, you have to login to each VM and use `"/bin/rm -f /var/aido/*"` command to remove `mongoConfig.cfg -*` files.

Step 11 On local site, apply the previous version of `mongoConfig.cfg` file by executing the following command .

```
/var/qps/install/current/scripts/build/build_etc.sh
```

Step 12 On remote site, apply the previous version of `mongoConfig.cfg` file by executing the following command.

```
/var/qps/install/current/scripts/build/build_etc.sh
```

Step 13 Verify the health check using `diagnostics.sh` command on local and remote site.

Replica Set Arbiter: Security

As arbiters do not replicate any data, including user/role details, they just participate when voting happens for electing new primary. Thus, when the authentication is enabled, the only way to login to them is through the localhost exception.

For more information, refer to https://docs.mongodb.com/v3.6/core/replica-set-arbiter/#security.

A few commands (such as, `isMaster`, `ping`, `connectionStatus`, and `authenticate`) do not require any authentication even if authentication is enabled. This is because these are used to support to connect to a deployment. On the other hand, majority of commands (including `rs.status()`, `show dbs`, `show collections` and so on) requires authentication if authentication is enabled.

For detailed command list, refer to https://docs.mongodb.com/v3.6/reference/command/

Admin Database

Purpose

By default, admin replica-set holds the following databases:

- **sharding:** This database holds the following information:
  - Session sharding: Session shard seeds and its databases.
  - Session type counters: Session statistics information. For example, number of sessions present in each shard for Gx, Rx, Sy, and so on.
  - Session compression dictionary data: Compression object data of session fields data.
  - Memcache rings data: Memcached rings and their sets data. Every set has two sessionmgr VMs followed with memcached port number.
  - Secondary Key sharding: Secondary Key shards seeds and its databases.
  - License data: Session license information.
- **scheduler:** This database holds the information about rebuilding the secondary key tasks data. When you execute `rebuild sk rings` or `rebuild sk db`, application creates the scheduled tasks to the “tasks” collection in this database. Once tasks are created, application pulls the information from the collection and execute those taks.
- **diameter:** This database holds the information about connected peers (inbound and outbound) connected to which load balance instance. This also holds the history of peers connected (start) and disconnected (stop) followed with timestamps.
- **queueing:** This database holds the information about internal TCP connection between the Policy Director (LB) and Policy Server (QNS) VMs in and out queue data.
- **clusters:** This database holds the information about sitenames and IP address of the ADMIN replica-set members in `hosts` collections.
- **policy_trace:** This database holds the information about the particular subscriber traces. To view the traces, you need to enable the trace for a particular subscriber.
- **Keystore:** This database holds the information about the redis key store configuration.

Note There are separate configurations available for the Trace Database and Endpoint Database in Cluster configuration under Policy Builder.

- If Trace Database is configured, "policy_trace" database is stored in configured replica-set.
- If Endpoint Database is configured, "diameter, and queueing" databases are stored in configured replica-set.

For more information on Trace Database and Endpoint Database configuration, see *Adding an HA Cluster* section in *CPS Mobile Configuration Guide*.

Note In GR deployment, if any site looses the connectivity (Internal/Replication) to the ADMIN replica-set, then the following impact is observed:

- If Admin/Endpoint databases are shared across all the sites.
  - There can be communication issue between the Policy Server (QNS) and Policy Director (LB) VMs. As the application is not able to get to the connected peers to send the processing messages, timeouts or drops can be observed.
  - Cross site messaging of stale session RARs from Policy Director (LB) of failed site to Policy Director (LB) of peer of running site fails.
- If Endpoint databases are not shared (co-located locally) across all the sites.
  - There is no communication issue between the Policy Server (QNS) and Policy Director (LB) VMs. As the application is able to get to the connected peers information locally to send the processing messages.
  - Cross site messaging of stale session RARs from Policy Director (LB) of failed site to Policy Director (LB) of peer running site fails and vice versa.
- Grafana does not display the session counters properly.
- If subscriber trace is enabled, the policy_trace cannot insert the trace information about the configured subscribers.

Protocol

Not applicable.

Port

Not applicable.

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.

CLI Access

Use the following command to access the OSGi console:

`telnet` <*ip*> <*port*>

The following commands can be executed on the OSGi console:

`ss` : List installed bundle status.

`start` <*bundle-id*> : Start the bundle.

`stop` <*bundle-id*> : Stop the bundle.

`diag` <*bundle-id*> : Diagnose the bundle.

Sharding Commands

Use the following OSGi commands to add or remove shards:

Table 2: Sharding Commands

Command	Description
`listshards`	Lists all the shards.
`removeshard <shard id>`	Marks the shard for removal. If shard is non-backup, rebalance is required for shard to be removed fully. If shard is backup, it does not require rebalance of sessions and hence would be removed immediately.
`rebalance <rate limit>`	Rebalances the buckets and migrates session with rate limit. Rate limit is optional. If rate limit is passed, it is applied at rebalance.
`rebalancebg <rate limit>`	Rebalances the buckets and schedules background task to migrate sessions. Rate limit is optional. If rate limit is passed, it is applied at rebalance.
`rebalancestatus`	Displays the current rebalance status. Status can be one of the following: • Rebalance is running (Remaining buckets: <pending count>) • Rebalance is required • Rebalanced

Command	Description
`rebuildAllSkRings`	In order for CPS to identify a stale session from the latest session, the secondary key mapping for each site stores the primary key in addition to the bucket ID and the site ID, that is, Secondary Key = <Bucket Id>; <Site Id>; <Primary Key>. To enable this feature, add the flag `-Dcache.config.version=1` in the `/etc/broadhop/qns.conf` file. Enabling this flag and running `rebuildAllSkRings` starts the data migration for the new version so that CPS can load the latest version of the session.
`skRingRebuildStatus`	Displays the status of the migration and the current cache version.
`listskshard`	List the SK shards.
`addskshard seed1[,seed2] port db-index [backup]`	Adds new SK shard. For backup shard, pass the backup option.
`removeskshard shardid [confirm]`	Mark SK shard for deletion.
`rebalancesk [rate limit]`	Rebalance SK buckets across SK shards in foreground.
`migratesk [rate limit]`	Migrate SK data in foreground. If data is already migrated it will query and skip.
`rebalanceskbg [rate limit]`	Rebalance SK buckets across SK shards and schedule the distribute task to migrate SK data in background on multiple QNS
`migrateskbg [rate limit]`	Schedule the distribute task to migrate SK data in background on multiple QNS. If data is already migrated it will query and skip.
`rebalanceskstatus`	Show SK DB shard rebalance status.
`rebuildskdb [rate limit]`	Rebuild SK DB from Session DB. Default rate limit is 1000.
`rebuildskdbstatus`	Show current SK DB rebuild status.
`getskorder`	Get current caching system priority order.
`setskorder skcache1 [skcache2]`	• Change secondary key caching system priority order • skcache1[2] can be MEMCACHE or SK_DB • skcache1 and skcache2 must not be same • skcache2 is optional, If skcache2 is not provided it will be disabled Example: • setskorder SK_DB: Enables SK_DB and disables MEMCACHE • setskorder MEMCACHE SK_DB: Enables MEMCACHE as PRIMARY and SK_DB as FALLBACK

Command	Description
`listskshard siteId`	Lists the SK shards for corresponding site ID.
`addskshard seed1[,seed2] port db-index siteid [backup]`	Adds new SK shard for mentioned site. For backup shard, add the backup option.
`rebalances siteid [rate limit]`	Rebalance SK buckets across SK shards in foreground for the mentioned site.
`migratesk siteid [rate limit]`	Migrate SK data in foreground for the mentioned site. If data is already migrated it will query and skip.
`rebalanceskbg siteid [rate limit]`	Rebalance SK buckets across SK shards for the mentioned site ID and schedule the distribute task to migrate SK data in background on multiple Policy Servers (QNS).
`migrateskbg siteid [rate limit]`	Schedule the distribute task to migrate SK data in background on multiple Policy Servers (QNS) for the mentioned site. If data is already migrated it will query and skip.
`rebalanceskstatus siteid`	Show SK database shard rebalance status for the mentioned site.
`rebuildskdb siteid [rate limit]`	Rebuild SK database from Session database for the mentioned site. Default rate limit is 1000.
`rebuildskdbstatus siteid`	Show current SK database rebuild status for the mentioned site ID.

CPS Alarm Commands

Use the following OSGi command to get the information related to open application alarms in CPS:

Table 3: Alarm Commands

Command	Description
`listalarms`	To list the open/active application alarms since last restart of policy server (QNS) process on pcrfclient01/02 VM.

Example:

```
osgi> listalarms
Active Application Alarms
id=1000 sub_id=3001  event_host=lb02  status=down date=2017-11-22,10:47:34,
051+0000  msg="3001:Host: site-host-gx Realm: site-gx-client.com is down"
id=1000 sub_id=3001  event_host=lb02  status=down date=2017-11-22,10:47:34,
048+0000  msg="3001:Host: site-host-sd Realm: site-sd-client.com is down"
id=1000 sub_id=3001  event_host=lb01  status=down date=2017-11-22,10:45:17,
927+0000  msg="3001:Host: site-server Realm: site-server.com is down"
id=1000 sub_id=3001  event_host=lb02  status=down date=2017-11-22,10:47:34,
091+0000  msg="3001:Host: site-host-rx Realm: site-rx-client.com is down"
id=1000 sub_id=3002  event_host=lb02  status=down date=2017-11-22,10:47:34,
111+0000  msg="3002:Realm: site-server.com:applicationId: 7:all peers are down"
```

Memcache Commands

Use the following OSGi commands to get the information related to memcache:

Note The memcache commands have been deprecated in CPS 20.1.0 and later releases.

Table 4: Memcache Commands

Command	Description
`disableCacheAudit`	Used to disable the complete memcached audit. Default: enable
`enableCacheAudit`	Used to enable the regular memcached audit. Default: enable
`cacheAuditStatus`	Used to display the current regular memcached audit status.
`enableFtsBasedCacheAudit`	Used to enable Full Table Scan (FTS) threshold based audit. This works only when audit feature is enabled. Default: true
`disableFtsBasedCacheAudit`	Used to disable the FTS Threshold based audit.
`ftsBasedCacheAuditStatus`	Used to display the current FTS based memcached audit status.
`currentCacheAuditInterval`	Used to display current periodic memcached audit interval.
`updateCacheAuditInterval` `Audi Interval[Integer]`	Used to update the regular memcached audit interval. Audit interval cannot be less than 360 minutes.
`currentFTSCacheAuditThreshold`	Used to provide the current FTS threshold for FTS based memcached audit.
`setFTSCacheAuditThreshold` `ThresholdVal[Integer]`	Used to specify the FTS threshold value for FTS based memcached audit. This value cannot be less than 25% of total allowed FTS per qns.
`nextCacheAuditSchedule`	Used to display next regular memcached audit schedule when memcache audit is done.

Ports

pcrfclientXX:

- Control Center: 9091
- Policy Builder: 9092

lbXX:

- iomanager: 9091
- Diameter Endpoints: 9092, 9093, 9094 ...

qnsXX: 9091

Ports should be blocked using a firewall to prevent access from outside the CPS cluster.

Accounts and Roles

Not applicable.

Policy Builder GUI

Purpose

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

URL and Port

HA: https://<*lbvip01*>:7443/pb

Protocol

HTTPS/HTTP

Accounts and Roles

Initial accounts are created during the software installation. Refer to the *CPS Operations Guide* for commands to add users and change passwords.

REST API

Purpose

To allow initial investigation into a Proof of Concept API for managing a CPS System and Custom Reference Data related through an HTTPS accessible JSON API.

CLI Access

This is an HTTPS/Web interface and has no Command Line Interface.

URL and Port

API: http://<*Cluster Manager IP*>:8458

Documentation: http://<*Cluster Manager IP*>:7070/doc/index.html

Accounts and Roles

Initial accounts are created during the software installation. Refer to the *CPS Operations Guide* for commands to add users and change passwords.

Rsyslog

Purpose

Enhanced log processing is provided using Rsyslog.

Rsyslog logs Operating System (OS) data locally (`/var/log/messages` etc.) using the `/etc/rsyslog.conf` and `/etc/rsyslog.d/*conf` configuration files.

rsyslog outputs all WARN level logs on CPS VMs to `/var/log/warn.log` file.

On all nodes, Rsyslog forwards the OS system log data to lbvip02 via UDP over the port defined in the logback_syslog_daemon_port variable as set in the CPS deployment template (Excel spreadsheet). To download the most current CPS Deployment Template (`/var/qps/install/current/scripts/deployer/templates/QPS_deployment_config_template.xlsm`), refer to the *CPS Installation Guide for VMware* or *CPS Release Notes* for this release.

Additional information is available in the Logging chapter of the *CPS Troubleshooting Guide*. Refer also to http://www.rsyslog.com/doc/ for the Rsyslog documentation.

CLI Access

Not applicable.

Protocol

UDP

Port

6514

Accounts and Roles

Account and role management is not applicable.

Rsyslog Customization

CPS provides the ability to configure forwarding of consolidated syslogs from rsyslog-proxy on Policy Director VMs to remote syslog servers (refer to *CPS Installation Guide for VMware*). However, if additional customizations are made to rsyslog configuration to forward logs to external syslog servers in customer's network for monitoring purposes, such forwarding must be performed via dedicated action queues in rsyslog. In the absence of dedicated action queues, when rsyslog is unable to deliver a message to the remote server, its main message queue can fill up which can lead to severe issues, such as, preventing SSH logging, which in turn can prevent SSH access to the VM.

Sample configuration for dedicated action queues is available in the *Logging* chapter of the *CPS Troubleshooting Guide*. Refer to rsyslog documentation on http://www.rsyslog.com/doc/v5-stable/concepts/queues.html for more details about action queues.

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

Use the following commands to access SVN:

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

Get all files from the server:

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

Example:

```
svn checkout --username broadhop --password broadhop
http://pcrfclient01/repos/configuration/root/configuration
```

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

Store/check-in the changed files to the server:

```
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/
```



Note Use `svn --help` for a list of other commands.

Protocol

HTTP

Port

80

Accounts and Roles

CPS 7.0 and Higher Releases

Add User with Read Only Permission

From the pcrfclient01 VM, run **adduser.sh** to create a new user.

```
/var/qps/bin/support/adduser.sh
```

Note This command can also be run from the Cluster Manager VM, but you must include the OAM (PCRFCLIENT) option:

```
/var/qps/bin/support/adduser.sh pcrfclient
```

Example:

```
[root@pcrfclient01 /]# /var/qps/bin/support/adduser.sh
Enter username: <username>
Enter group for the user: <any group>
Enter password:
Re-enter password:
```

Add User with Read/Write Permission

By default, the **adduser.sh** script creates a new user with read-only permissions. For read-write permission, you must assign the user to the **qns-svn** group and then run the **vm-init** command.

From the pcrfclient01 VM, run the **adduser.sh** script to create the new user.

Run the following command on both pcrfclient01 and pcrfclient02 VMs:

```
/etc/init.d/vm-init
```

You can now login and commit changes as the newly created user.

Change Password

From the pcrfclient01 VM, run the **change_passwd.sh** script to change the password of a user.

```
/var/qps/bin/support/change_passwd.sh
```

Example:

```
[root@pcrfclient01 /]# /var/qps/bin/support/change_passwd.sh
Enter username whose password needs to be changed: user1
Enter current password:
Enter new password:
Re-enter new password:
```

CPS Versions Earlier than 7.0

Perform all of the following commands on both the pcrfclient01 and pcrfclient02 VMs.

Add User

Use the htpasswd utility to add a new user

```
htpasswd -mb /var/www/svn/.htpasswd <username> <password>
```

Example:

```
htpasswd -mb /var/www/svn/.htpasswd user1 password
```

In some versions, the password file is `/var/www/svn/password`

Provide Access

Update the user role file /var/www/svn/users-access-file and add the username under `admins` (for read/writer permissions) or `nonadmins` (for read-only permissions). For example:

```
[groups]
admins = broadhop
nonadmins = read-only, user1
[/]
@admins = rw
@nonadmins = r
```

Change Password

Use the htpasswd utility to change passwords.

```
htpasswd -mb /var/www/svn/.htpasswd <username> <password>
```

Example:

```
htpasswd -mb /var/www/svn/.htpasswd user1 password
```

TACACS+ Interface

Purpose

CPS 7.0 and above has been designed to leverage the Terminal Access Controller Access Control System Plus (TACACS+) to facilitate centralized management of users. Leveraging TACACS+, the system is able to provide system-wide authentication, authorization, and accounting (AAA) for the CPS system.

Further the system allows users to gain different entitlements based on user role. These can be centrally managed based on the attribute-value pairs (AVP) returned on TACACS+ authorization queries.

CLI Access

No CLI is provided.

Port

CPS communicates to the AAA backend using IP address/port combinations configured by the operator.

Account Management

Configuration is managed by the Cluster Management VM which deploys the `/etc/tacplus.conf` and various PAM configuration files to the application VMs. For more account management information, refer to TACACS+ Service Requirements, on page 45.

For more information about TACACS+, refer to the following links:

- TACAC+ Protocol Draft: http://tools.ietf.org/html/draft-grant-tacacs-02

- Portions of the solution reuse software from the open source pam_tacplus project hosted at: https://github.com/jeroennijhof/pam_tacplus

For information on CLI commands, refer to Accessing the CPS CLI, on page 30.

Unified API

Purpose

Unified APIs are used to reference customer data table values.

URL and Port

HA: https://<*lbvip01*>:8443/ua/soap

Protocol

HTTPS/HTTP

Accounts and Roles

Currently there is no authorization for this API

Accessing the CPS CLI

sudo supports a plugin architecture for security policies and input/output logging. The default security policy is *sudoers*, which is configured via the file `/etc/sudoers`, contains the rules that users must follow when using the sudo command.

sudo allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments.

For example: `%adm ALL=(ALL) NOPASSWD: ALL`

This means that any user in the administrator group on any host may run any command as any user without a password. The first ALL refers to hosts, the second to target users, and the last to allowed commands.

When an authenticated user has one of the above group permissions, they can access the CPS CLI and run predefined commands available to that user role. A list of commands available after authentication can be viewed using the `sudo -l` command (-l for list), or any user with root privileges can use `sudo -l -U <qns-role>` to see the available command for a specific Policy Server (qns) role.

The `/etc/sudoers` file contains user specifications that define the commands that users may execute. When sudo is invoked, these specifications are checked in order, and the last match is used. A user specification looks like this at its most basic:

```
User Host = (Runas) Command
```

Read this as "User may run Command as the Runas user on Host". Any or all of the above may be the special keyword ALL, which always matches. User and Runas may be usernames, group names prefixed with %, numeric UIDs prefixed with #, or numeric GIDs prefixed with %#. Host may be a hostname, IP address, or a whole network (for example, 192.0.2.0/24), but not 127.0.0.1.

Group Identifiers

gid

The group identifier of the TACACS+ authenticated user on the VM nodes. This value should reflect the role assigned to a given user, based on the following values:

- group id=500 (qns)

  The group identifier used by Policy Server (qns) user in application.

- group id=501 (qns-su)

  This group identifier should be used for users that are entitled to attain superuser (or 'root') access on the CPS VM nodes.

- group id=504 (qns-admin)

  This group identifier should be used for users that are entitled to perform administrative maintenance on the CPS VM nodes.



Note To execute administrative scripts from qns-admin, prefix the command with `sudo`. For example

```
sudo stopall.sh
```

- group id=505 (qns-ro)

  This group identifier should be used for users that are entitled to read-only access to the CPS VM nodes.

When an authenticated user has one of the above group permissions, they can access the CPS CLI and run predefined commands available to that user role. A list of commands available after authentication can be viewed using the `sudo -l` command (-l for list), or any user with root privileges can use `sudo -l -U <qns-role>` to see the available command for a specific Policy Server (qns) role.

For more information, refer to https://www.sudo.ws/intro.html.

home

The user's home directory on the CPS VM nodes. To enable simpler management of these systems, the users should be configured with a pre-deployed shared home directory based on the role they are assigned with the gid.

- home=/home/qns-su should be used for users in the 'qns-su' group (gid=501)
- home=/home/qns-admin should be used for users in the 'qnsadmin' group (gid=504)
- home=/home/qns-ro should be used for users in the 'qns-ro' group (gid=505)

Support for Multiple User Login Credentials

CPS supports multiple user login credentials with different privileges for all non-cluman vms.

Add `allow_user_for_cluman` flag in configuration.csv file, to update sudoers file. This flag functionality supports the following different privileges accessible for cluman.

- When `allow_user_for_cluman` flag is set to true, sudoers file is updated with CPS users and they are able to access cluman according to their privilege.
- When `allow_user_for_cluman` flag is set to false or not defined, CPS users are not able to execute any commands from cluman.

The following table describes CSV based configuration parameters.

Table 5: CSV Based Configuration Parameters

Parameters	Description
allow_user_for_cluman	Used to update the `/etc/sudoers` with CPS entries on cluman.

This feature is supported only in VMware.

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.

This section covers the following topics:

- Create Users, on page 32
- Revert Configuration, on page 33

Create Users

Procedure

Step 1 Log in to the Cluster Manager.

Step 2 Add a user to CPS by executing:

```
adduser.sh
```

Step 3 When prompted for the user's group, set 'qns-svn' for read-write permissions or 'qns-ro' for read-only permissions.

- To check if a user already exists, login in as root and enter `su username`.
- To check a user's 'groups', enter `groups username`.
- To change a user's password, use the `change_passwd.sh` command.

Note
The `change_passwd.sh` script changes the password on all the VMs temporarily. You also need to generate an encrypted password. To generate encrypted password, refer to *System Password Encryption* in *CPS Installation Guide for VMware*. The encrypted password must be added in the `Configuration.csv` spreadsheet. To make

the new password persisent, execute `import_deploy.sh`. If the encrypted password is not added in the spreadsheet and `import_deploy.sh` is not executed, then after running `reinit.sh` script, the qns-svn user takes the existing default password from `Configuration.csv` spreadsheet.

Refer to CPS Commands for more information about these commands.

Revert Configuration

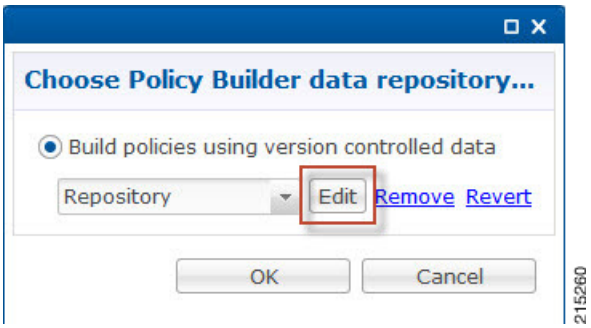
The user can revert the configuration if changes since the last publish/save to client repository are not wanted.

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.

Procedure

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.

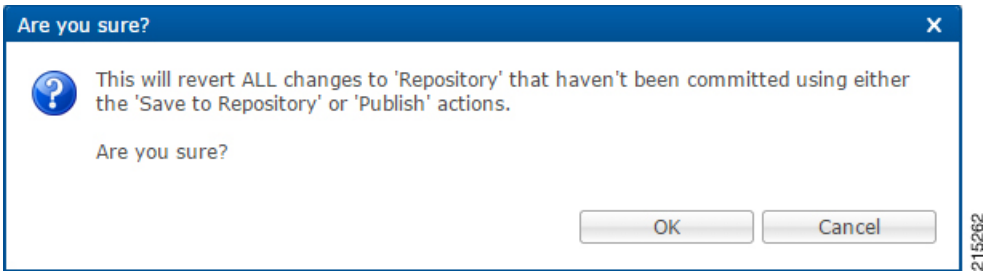
Figure 1: Verifying the User



Step 2 Click **Revert**.

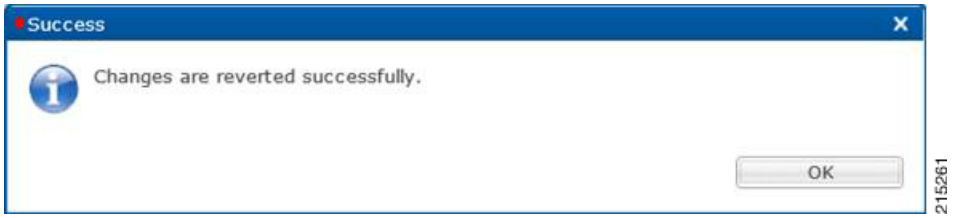
The following confirmation dialog opens.

Figure 2: 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 3: 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. Refer to the *CPS Mobile Configuration Guide* for steps to publish data to the 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://pcrfclient01/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.

Note Publishing will also do a Save to Client Repository to ensure the Policy Builder and Server configurations are not out of sync.

The default URL is http://pcrfclient01/repos/run.

Control Center Access

After the installation is complete, you need to configure the Control Center access. This is designed to give the customer a customized Control Center username.

Note Tacacs+ users can access control center by enabling TACACS authentication along with `tacacs_on_ui` flag.

Add a Control Center User

Procedure

Step 1 Login to the Cluster Manager VM.

Step 2 Execute the following script to add a Control Center user.

```
/var/qps/bin/support/adduser.sh
```

Note
To add a user with 'read/write' access to Control Center, their group should be 'qns'. To add a user with 'read' access to Control Center, their group should be 'qns-ro'.

Example:

```
/var/qps/bin/support/adduser.sh
Enter username: username
Enter group for the user: groupname
Enter password: password
Re-enter password: password
```

This example adds *username* to all the VMs in the cluster.

Update Control Center Mapping

This section describes updating Control Center mapping of read-write/read-only to user groups (Default: qns and qns-ro respectively).

Procedure

Step 1 Login to the Cluster Manager VM.

Step 2 Update `/etc/broadhop/authentication-provider.xml` to include the group mapping for the group you want to use.

Note
Make sure that this group exists on at least the Policy Server (QNS) VMs or adding users will fail due to no group available (there should be an entry in `/etc/group`).

In the following example, the 'test' group has been added as an read-write mapping for Control Center - updated line in bold:

```
<beans:beans xmlns="http://www.springframework.org/schema/security"
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
classpath:/org/springframework/beans/factory/xml/spring-beans-3.0.xsd
  http://www.springframework.org/schema/security
classpath:/org/springframework/security/config/spring-security-3.0.xsd">
<beans:bean id="authenticationProvider"
```

```
class="com.broadhop.ui.security.server.pam.PamAuthenticationProvider">
    <!-- change the key value to be the customer's role that maps to the cisco role. -->
    <beans:property name="roleMap">
      <beans:map>
        <beans:entry key="qns" value="ROLE_SUMADMIN" />
        <beans:entry key="test" value="ROLE_SUMADMIN" />
        <beans:entry key="qns-ro" value="ROLE_READONLY" />
      </beans:map>
    </beans:property>
  </beans:bean>

  <authentication-manager>
    <authentication-provider ref="authenticationProvider" />
  </authentication-manager>

</beans:beans>
```

Step 3 Run `syncconfig.sh` to put this file on all VMs.

Step 4 Restart the CPS system, so that the changes done above are reflected in the VMs:

```
restartall.sh
```

Caution
Executing `restartall.sh` will cause messages to be dropped.

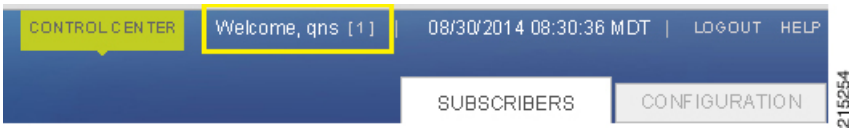
To add a new user to Control Center and specify the group you have specified in the configuration file above, refer to Add a Control Center User, on page 35.

Multiple Concurrent User Sessions

CPS Control Center supports session limits per user. If the user exceeds the configured session limit, they are not allowed to log in. CPS also provides notifications to the user when other users are already logged in.

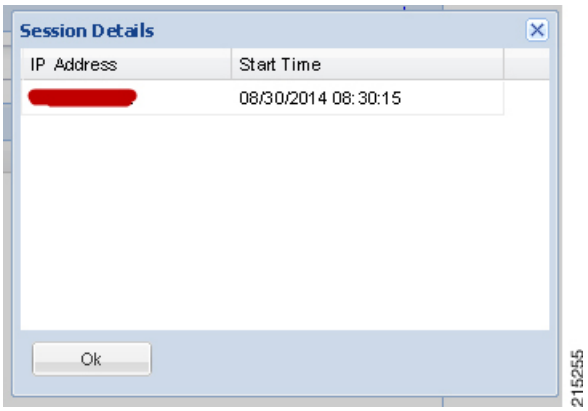
When a user logs in to Control Center, a Welcome message displays at the top of the screen. A session counter is shown next to the username. This represents the number of login sessions for this user. In the following example, this user is logged in only once ([1]).

Figure 4: Welcome Message



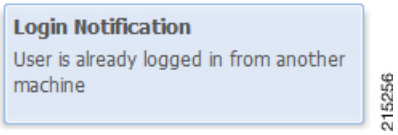
The user can click the session counter ([1]) link to view details for the session(s), as shown below.

Figure 5: Viewing Session Details



When another user is already logged in with the same username, a notification displays for the second user in the bottom right corner of the screen, as shown below.

Figure 6: Login Notification for a Second User



The first user also receives a notification, as shown, and the session counter is updated to [2].

Figure 7: Login Notification for First User

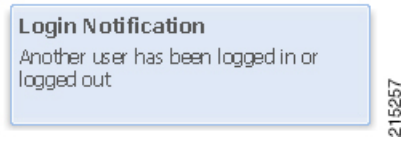
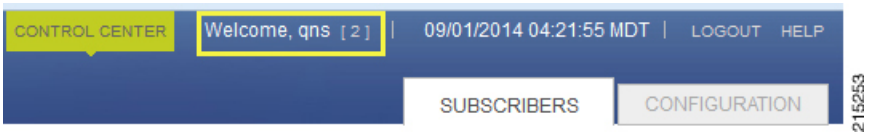


Figure 8: Indication of Two Users with Same Username



These notifications are not displayed in real time; CPS updates this status every 30 seconds.

Configure Session Limit

The session limit can be configured by the runtime argument, which can be configured in the qns.conf file.

-Dcc.user.session.limit=3 (default value is 5)

Configure Session Timeout

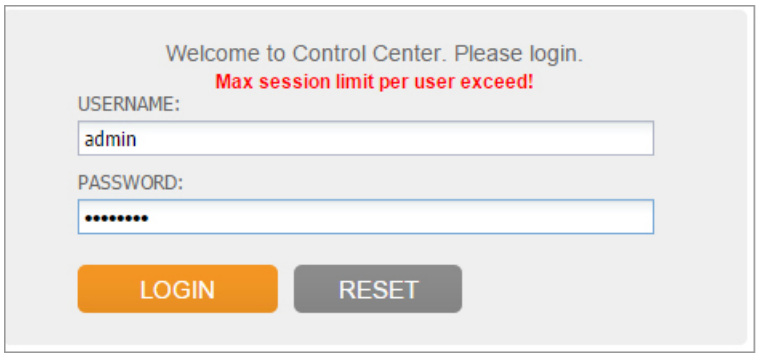
The default session timeout can be changed by editing the following file on the Policy Server (QNS) instance:

```
./opt/broadhop/qns-1/plugins/com.broadhop.ui_3.5.0.release/war/WEB-INF/web.xml

<!-- timeout after 15 mins of inactivity -->
<session-config>
<session-timeout>15</session-timeout>
    <cookie-config>
<http-only>true</http-only>
</cookie-config>
</session-config>
```

Note The same timeout value must be entered on all Policy Server (QNS) instances.

When the number of sessions of the user exceeds the session limit, the user is not allowed to log in and receives the message "`Max session limit per user exceed!`"



Important Notes

If a user does not log out and then closes their browser, the session remains alive on the server until the session times out. When the session timeout occurs, the session is deleted from the memcached server. The default session timeout is 15 minutes. This is the idle time after which the session is automatically deleted.

When a Policy Server (QNS) instance is restarted, all user/session details are cleared.

When the memcached server is restarted without also restarting the Policy Server (QNS) instance, all http sessions on the Policy Server (QNS) instance are invalidated. In this case the user is asked to log in again and after that, the new session is created.

Enable Authentication for Unified API

HAProxy is used to secure and balance calls to the CPS Unified API.

Procedure

Step 1 Back up the `/etc/haproxy/haproxy.cfg` file before making modifications in the following steps.

Step 2 Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 and add a userlist with at least one username and password.

Use the following syntax:

```
userlist <userlist name>

user <username1> password <encrypted password>
```

Use the following steps to generate a password hash:

a. Execute `/var/qps/install/current/scripts/bin/support/generate_encrypted_password.sh` script to get encrypted password.

b. After script execution the encrypted password will be like below.

```
+-----------------------------------------------------------------------------------------------------------------+
| Fri May 29 11:43:47 UTC 2020
                              |
| Encrypted key
                                          |
|
$6$bc732ffd2a5ad85e$dYuQfGowAsAS6E2mQyWgGtcSUY4IKss11.4AY1u852gGwZzr4Y54rBdkHG6zQytFPXXDJGwknx.IYIeDeW.jP.

          |
+-----------------------------------------------------------------------------------------------------------------+
```

Step 3 Run the following command to generate an encrypted password:

```
openssl passwd -1 <your-password>
```

Example:

```
openssl passwd -1 'password'
$1$u/fEW7/p$e//MBCq7AHQArNo61Qvy20
```

Step 4 Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 to configure HAProxy to require authentication. Add the following 4 lines to the `haproxy.cfg` file:

```
acl validateAuth http_auth(<userlist_name>)
acl unifiedAPI path_beg -i /ua/soap
http-request allow if !unifiedAPI
http-request auth unless validateAuth
```

The userlist created in Step 2, on page 38 needs to be mapped with the acl in the following line:

```
acl validateAuth http_auth(<userlist name>)
```

For example:

```
frontend https-api
description Unified API
bind lbvip01:8443 ssl crt /etc/ssl/certs/quantum.pem
default_backend api_servers
reqadd X-Forwarded-Proto:\ https if { ssl_fc }
backend api_servers
mode http
balance roundrobin
option httpclose
option abortonclose
option httpchk GET /ua/soap/keepalive
server qns01_A qns01:8080 check inter 30s
server qns02_A qns02:8080 check inter 30s
server qns03_A qns03:8080 check inter 30s
server qns04_A qns04:8080 check inter 30s
acl validateAuth http_auth(L1)
acl unifiedAPI path_beg -i /ua/soap
```

```
http-request allow if !unifiedAPI
http-request auth unless validateAuth
```

The configuration above applies authentication on context /ua/soap, which is the URL path of the Unified API.

Note
The `haproxy.cfg` file is generated by the Puppet tool. Any manual changes to the file in lb01/lb02 would be reverted if the pupdate or vm-init scripts are run.

Unified API Security: Access Privileges

By default, the CPS Unified API does not require username and password authentication. To enable authentication, refer to Enable Authentication for Unified API.

There are two options to include a username and password in an API request:

- Include the username and password directly in the request. For example:

  https://<username>:<password>@<lbvip02>:8443/ua/soap

- Add an authentication header to the request:

  `Authorization: Basic <base64 encoded value of username:password>`

  For example:

  ```
  wget -d -O - --header="Authorization: Basic cG9ydGFXNjbzEyMwo="
  https://lbvip02:8443/ua/soap/keepalive
  ```

Enable Authentication for Unified API

HAProxy is used to secure and balance calls to the CPS Unified API.

Procedure

Step 1 Back up the `/etc/haproxy/haproxy.cfg` file before making modifications in the following steps.

Step 2 Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 and add a userlist with at least one username and password.

Use the following syntax:

`userlist <userlist name>`

`user <username1> password <encrypted password>`

Use the following steps to generate a password hash:

a. Execute `/var/qps/install/current/scripts/bin/support/generate_encrypted_password.sh` script to get encrypted password.

b. After script execution the encrypted password will be like below.

```
+-------------------------------------------------------------------------------------------------------------+
```

```
| Fri May 29 11:43:47 UTC 2020
                              |
| Encrypted key
                                          |
|
$6$bc732ffd2a5ad85e$dYuQfGowAsAS6E2mQyWgGtcSUY4IKss11.4AY1u852gGwZzr4Y54rBdkHG6zQytFPXXDJGwknx.IYIeDeW.jP.

          |
+-------------------------------------------------------------------------------------------------------------+
```

Step 3 Run the following command to generate an encrypted password:

```
openssl passwd -1 <your-password>
```

Example:

```
openssl passwd -1 'password'
$1$u/fEW7/p$e//MBCq7AHQArNo61Qvy20
```

Step 4 Edit `/etc/haproxy/haproxy.cfg` on lb01/lb02 to configure HAProxy to require authentication. Add the following 4 lines to the `haproxy.cfg` file:

```
acl validateAuth http_auth(<userlist_name>)
acl unifiedAPI path_beg -i /ua/soap
http-request allow if !unifiedAPI
http-request auth unless validateAuth
```

The userlist created in Step 2, on page 40 needs to be mapped with the acl in the following line:

```
acl validateAuth http_auth(<userlist name>)
```

For example:

```
frontend https-api
description Unified API
bind lbvip01:8443 ssl crt /etc/ssl/certs/quantum.pem
default_backend api_servers
reqadd X-Forwarded-Proto:\ https if { ssl_fc }
backend api_servers
mode http
balance roundrobin
option httpclose
option abortonclose
option httpchk GET /ua/soap/keepalive
server qns01_A qns01:8080 check inter 30s
server qns02_A qns02:8080 check inter 30s
server qns03_A qns03:8080 check inter 30s
server qns04_A qns04:8080 check inter 30s
acl validateAuth http_auth(L1)
acl unifiedAPI path_beg -i /ua/soap
http-request allow if !unifiedAPI
http-request auth unless validateAuth
```

The configuration above applies authentication on context /ua/soap, which is the URL path of the Unified API.

Note
The `haproxy.cfg` file is generated by the Puppet tool. Any manual changes to the file in lb01/lb02 would be reverted if the pupdate or vm-init scripts are run.

WSDL and Schema Documentation

In order to access the Unified API WSDL while using authentication change the following line:

```
acl unifiedAPI path_beg -i /ua/soap
```

to

```
acl unifiedAPI path_beg -i /ua/.
```

The default address for the WSDL is `https://<lbvip01>:8443/ua/wsdl/UnifiedApi.wsdl`

The Unified API contains full documentation in an html format that is compatible with all major browsers.

The default address is `https://<HA-server-IP>:8443/ua/wsdl/UnifiedApi.xsd`



Note Run the **about.sh** command from the Cluster Manager to display the actual addresses as configured in your deployment.

Enabling Unified API Access on HTTP Port 8080

CPS 7.x onward uses HTTPS on port 8443 for Unified API access. To enable HTTP support (like pre-7.0) on port 8080, perform the following steps:



Note Make sure to open port 8080 if firewall is used on the setup.

Procedure

Step 1 Create the following directories (ignore File exists error), on Cluster Manager:

```
/bin/mkdir -p /var/qps/env_config/modules/custom/templates/etc/haproxy
/bin/mkdir -p /var/qps/env_config/modules/custom/templates/etc/monit.d
/bin/mkdir -p /var/qps/env_config/nodes
```

Step 2 Create the file `/var/qps/env_config/modules/custom/templates/etc/haproxy/haproxy-soaphttp.erb` with the following contents on Cluster Manager:

- Change XXXX with the Unified API interface hostname or IP
- In this example, we are adding 10 Policy Servers (QNS). You can add/remove the number of Policy Servers (QNS) depending on your network requirements.

```
global
  daemon
  nbproc       1          # number of processing cores
  stats socket /tmp/haproxy-soaphttp
defaults
  timeout client    60000ms       # maximum inactivity time on the client side
```

```
  timeout server    180000ms      # maximum inactivity time on the server side
  timeout connect   60000ms       # maximum time to wait for a connection attempt to a server to
succeed

  log             127.0.0.1       local1 err

listen pcrf_proxy XXXX:8080 ----------- > where, XXXX, is Unified API interface hostname or IP
        mode http
        balance roundrobin
        option httpclose
        option abortonclose
        option httpchk GET /ua/soap/KeepAlive
        server qns01_A qns01:8080 check inter 30s
        server qns02_A qns02:8080 check inter 30s
        server qns03_A qns03:8080 check inter 30s
        server qns04_A qns04:8080 check inter 30s
        server qns05_A qns05:8080 check inter 30s
        server qns06_A qns06:8080 check inter 30s
        server qns07_A qns07:8080 check inter 30s
        server qns08_A qns08:8080 check inter 30s
        server qns09_A qns09:8080 check inter 30s
        server qns10_A qns10:8080 check inter 30s
```

Step 3 Create the file `/var/qps/env_config/modules/custom/templates/etc/monit.d/haproxy-soaphttp` with the following contents on Cluster Manager:

```
check process haproxy-soaphttp with pidfile /var/run/haproxy-soaphttp.pid
start = "/usr/bin/systemctl start haproxy-soaphttp"
stop = "/usr/bin/systemctl stop haproxy-soaphttp"
```

Step 4 Create or modify the `/var/qps/env_config/nodes/lb.yaml` file with the following contents on Cluster Manager:

If the file exists then just add `custom::soap_http:`

```
classes:
  qps::roles::lb:
  custom::soap_http:
```

Step 5 Create the file `/var/qps/env_config/modules/custom/manifests/soap_http.pp` with the following contents on Cluster Manager.

Change ethX with the Unified API IP interface like eth0/eth1/eth2.

```
class custom::soap_http(
  $haproxytype = "-soaphttp",
)
{
  service { "haproxy-soaphttp":
    enable => false,
    require => [Package [ "haproxy" ],File ["/etc/haproxy/haproxy-soaphttp.cfg"],
File['/etc/init.d/haproxy-soaphttp'], Exec["sysctl_refresh"]],
  }
  file { "/etc/init.d/haproxy-soaphttp":
    owner => "root",
    group => "root",
    content => template('qps/etc/init.d/haproxy'),
    require => Package [ "haproxy" ],
    notify => Service['haproxy-soaphttp'],
    mode => 0744
  }
  file { "/etc/haproxy/haproxy-soaphttp.cfg":
    owner => "root",
    group => "root",
    content => template('custom/etc/haproxy/haproxy-soaphttp.erb'),
```

```
    require => Package [ "haproxy" ],
    notify => Service['haproxy-soaphttp'],
  }
  file { "/etc/monit.d/haproxy-soaphttp":
    content => template("custom/etc/monit.d/haproxy-soaphttp"),
    notify => Service["monit"],
  }
  exec { "remove ckconfig for haproxy-soaphttp":
    command => "/sbin/chkconfig --del haproxy-soaphttp",
    require => [Service['haproxy-soaphttp']],
  }
  firewall { '100 allow soap http':
    port => 8080,
    iniface => "ethX",
    proto => tcp,
    action => accept,
  }
}
```

Step 6 Validate the syntax of your newly created Puppet script on Cluster Manager:

```
/usr/bin/puppet parser validate /var/qps/env_config/modules/custom/manifests/soap_http.pp
```

Step 7 Rebuild your Environment Configuration on Cluster Manager:

```
/var/qps/install/current/scripts/build/build_env_config.sh
```

Step 8 Reinitialize your lb01/02 environments on Cluster Manager:

The following commands will take few minutes to complete.

```
ssh lb01 /etc/init.d/vm-init
ssh lb02 /etc/init.d/vm-init
```

Step 9 Validate SOAP request on http:

a) Verify the haproxy services are running on lb01 and lb02 by executing the commands on Cluster Manager:

```
ssh lb01 monit summary | grep haproxy-soaphttp
Process 'haproxy-soaphttp'                  Running
ssh lb01 service haproxy-soaphttp status
haproxy (pid  11061) is running...
ssh lb02 monit summary | grep haproxy-soaphttp
Process 'haproxy-soaphttp'                  Running
ssh lb02 service haproxy-soaphttp status
haproxy (pid  13458) is running...
```

b) Verify the following URLs are accessible:

```
Unified API WSDL: http://<IP address>:8080/ua/wsdl/UnifiedApi.wsdl
Unified API XSD: http://<IP address>:8080/ua/wsdl/UnifiedApi.xsd
```

where, <*IP address*> is the IP address set in Step 2, on page 42.

TACACS+

This section covers the following topics:

• Overview, on page 45
• TACACS+ Service Requirements, on page 45
• Caching of TACACS+ Users, on page 46

Overview

Cisco Policy Suite (CPS) is built around a distributed system that runs on a large number of virtualized nodes. Previous versions of the CPS software allowed operators to add custom accounts to each of these virtual machines (VM), but management of these disparate systems introduced a large amount of administrative overhead.

CPS has been designed to leverage the Terminal Access Controller Access Control System Plus (TACACS+) to facilitate centralized management of users. Leveraging TACACS+, the system is able to provide system-wide authentication, authorization, and accounting (AAA) for the CPS system.

Further the system allows users to gain different entitlements based on user role. These can be centrally managed based on the attribute-value pairs (AVP) returned on TACACS+ authorization queries.

TACACS+ Service Requirements

To provide sufficient information for the Linux-based operating system running on the VM nodes, there are several attribute-value pairs (AVP) that must be associated with the user on the ACS server used by the deployment. User records on Unix-like systems need to have a valid “passwd” record for the system to operate correctly. Several of these fields can be inferred during the time of user authentication, but the remaining fields must be provided by the ACS server.

A standard “passwd” entry on a Unix-like system takes the following form:

```
<username>:<password>:<uid>:<gid>:<gecos>:<home>:<shell>
```

When authenticating the user via TACACS+, the software can assume values for the username, password, and gecos fields, but the others must be provided by the ACS server. To facilitate this need, the system depends on the ACS server provided these AVP when responding to a TACACS+ Authorization query for a given username:

- uid

  A unique integer value greater than or equal to 501 that serves as the numeric user identifier for the TACACS+ authenticated user on the VM nodes. It is outside the scope of the CPS software to ensure uniqueness of these values.

- gid

  The group identifier of the TACACS+ authenticated user on the VM nodes. This value should reflect the role assigned to a given user, based on the following values:

  - `gid=501 (qns-su)`

    This group identifier should be used for users that are entitled to attain superuser (or 'root') access on the CPS VM nodes.

  - `gid=504 (qns-admin)`

    This group identifier should be used for users that are entitled to perform administrative maintenance on the CPS VM nodes.

Note For stopping/starting the Policy Server (QNS) process on node, the qns-admin user should use `monit`:

For example,

```
sudo monit stop qns-1
sudo monit start qns-1
```

- `gid=505 (qns-ro)`

  This group identifier should be used for users that are entitled to read-only access to the CPS VM nodes.

- home

  The user's home directory on the CPS VM nodes. To enable simpler management of these systems, the users should be configured with a pre-deployed shared home directory based on the role they are assigned with the gid.

  - home=/home/qns-su should be used for users in the `qns-su` group (gid=501)
  - home=/home/qns-admin should be used for users in the `qnsadmin` group (gid=504)
  - home=/home/qns-ro should be used for users in the `qns-ro` group (gid=505)

- shell

  The system-level login shell of the user. This can be any of the installed shells on the CPS VM nodes, which can be determined by reviewing the contents of `/etc/shells` on one of the CPS VM nodes. Typically, this set of shells is available in a CPS deployment:

  - /bin/sh
  - /bin/bash
  - /sbin/nologin
  - /bin/dash
  - /usr/bin/sudosh

  The `/usr/bin/sudosh` shell can be used to audit user's activity on the system.

Caching of TACACS+ Users

The user environment of the Linux-based VMs needs to be able to lookup a user's `passwd` entry via different columns in that record at different times. The TACACS+ NSS module provided as part of the CPS solution however is only able to query the Access Control Server (ACS) for this data using the `username`. For this reason the system relies upon the Name Service Cache Daemon (NSCD) to provide this facility locally after a user has been authorized to use a service of the ACS server.

More details on the operations of NSCD can be found by referring to online help for the software (nscd --help) or in its man page (nscd(8)). Within the CPS solution it provides a capability for the system to lookup a user's `passwd` entry via their `uid` as well as by their `username`.

To avoid cache coherence issues with the data provided by the ACS server the NSCD package has a mechanism for expiring cached information.

The default NSCD package configuration on the CPS VM nodes has the following characteristics:

- Valid responses from the ACS server are cached for 600 seconds (10 minutes)
- Invalid responses from the ACS server (user unknown) are cached for 20 seconds
- Cached valid responses are reloaded from the ACS server 5 times before the entry is completely removed from the running set -- approximately 3000 seconds (50 minutes)
- The cache are persisted locally so it survives restart of the NSCD process or the server

It is possible for an operator to explicitly expire the cache from the command line. To do so the administrator need to get the shell access to the target VM and execute the following command as a root user:

```
# nscd -i passwd
```

The above command will invalidate all entries in the passwd cache and force the VM to consult with the ACS server for future queries.

There may be some unexpected behaviors of the user environment for TACACS+ authenticated users connected to the system when their cache entries are removed from NSCD. This can be corrected by the user by logging out of the system and logging back into it or by issuing the following command, which forces the system to query the ACS server:

```
# id -a "$USER"
```

Reading Log Files

Only qns-ro and qns-admin users are allowed to view log files at specific paths according to their role and maintenance requirement. Access to logs are allowed only using the following paths:

- /var/log/
- /var/log/broadhop/scripts/
- /var/log/httpd
- /var/log/redis
- /var/log/broadhop

Commands such as `cat`, `less`, `more`, and `find` cannot be executed using sudo in CPS 10.0.0 or higher releases.

To read any file, execute the following script using sudo:

```
$ sudo /var/qps/bin/support/logReader.py -r h -n 2 -f /var/log/puppet.log
```

where,

- `-r`: Corresponds to tail (t) ,tailf (tf), and head (h) respectively
- `-n`: Determines number of lines to be read. It works with the -r option. This is an optional parameter.
- `-f`: Determines the complete file path to be read.

Note

- Non-root users cannot view the sudosh logs.
- Support to read gunzipped files is also available.

CRD APIs

You 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.

Table names must be all in lowercase alphanumeric to utilize these APIs. Neither spaces nor special characters are allowed in the table name.

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

Setup Requirements

Policy Server

The feature com.broadhop.custrefdata.service.feature needs to be installed on the Policy Server.

In a High Availability (HA)/Distributed CPS deployment, this feature should be installed on the QNS0x nodes.

Policy Builder

The feature `com.broadhop.client.feature.custrefdata` needs to be installed in Policy Builder.

Procedure

Step 1 Login into Policy Builder.
Step 2 Select **Reference Data** tab.
Step 3 From the left pane, select **Systems**.
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.

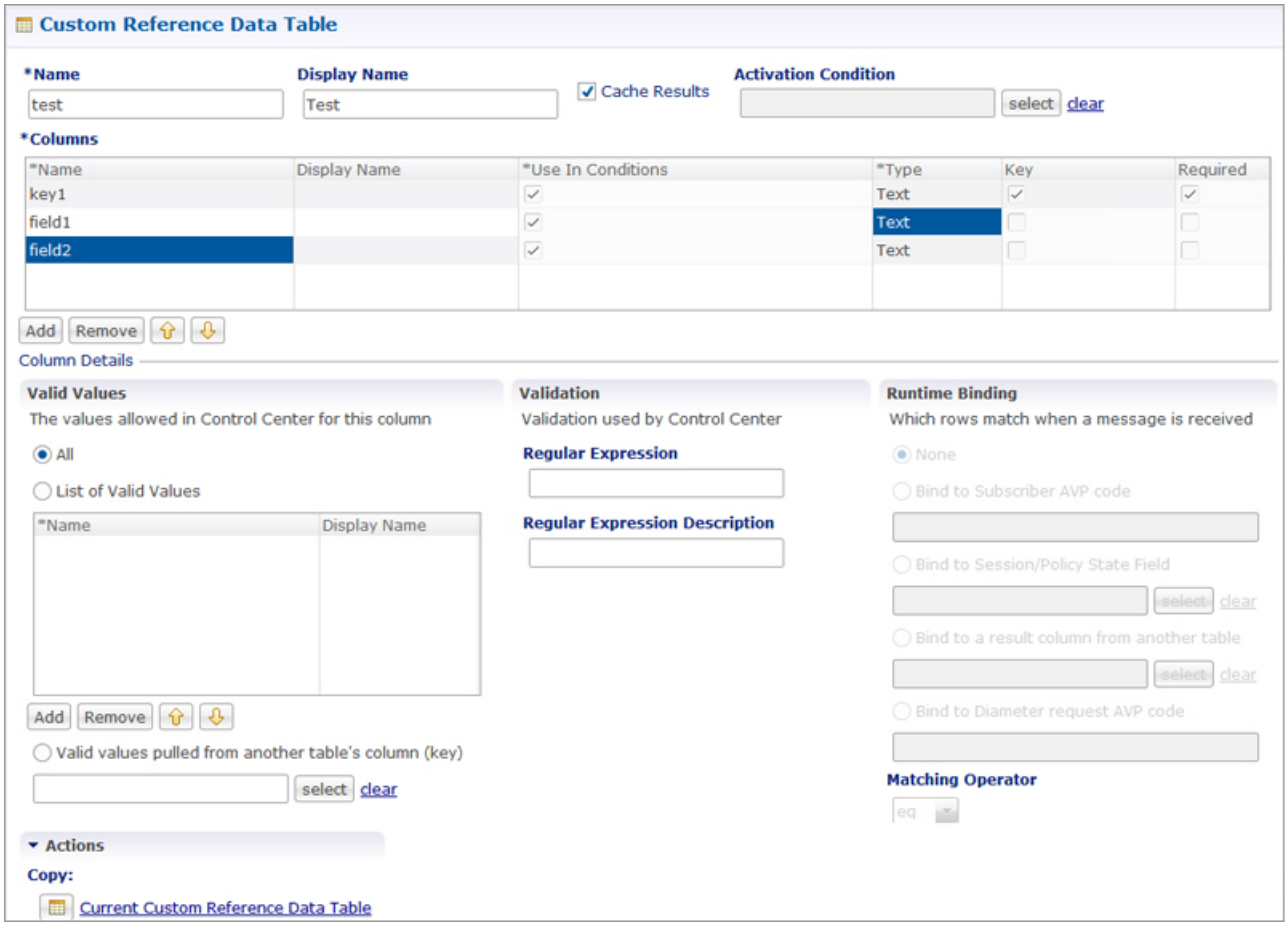
The following parameters can be configured under **Custom Reference Data Configuration**:

Table 6: Custom Reference Data Configuration

Parameter	Description
Primary Database IP Address	IP address of the primary sessionmgr database.
Secondary Database IP Address	Optional, this field is the IP address of a secondary, backup, or failover sessionmgr database.
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 Reference Data Tables**, at least one Custom Reference Data Table must be defined.

Figure 9: Custom Reference Data Table



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

Table 7: Custom Reference Data Table Parameters

Parameter	Description
Name	This is the name of the table that will be stored in the database. It should start with alphanumeric characters, should be lowercase OR uppercase but not MixedCase, and should not start with numbers, no special characters are allowed, use "_" to separate words. For example, logical_apn = GOOD, logicalAPN = BAD, no_spaces. For more information, refer to Limitations, on page 48.
Display Name	This is the name of the table that will be displayed in Control Center.
Cache Results	This indicates whether the tables should be cached in memory. This should be checked for production. For more information, refer to Caching, on page 53.

Parameter	Description
Activation Condition	This is the Custom Reference Data Trigger which needs to be true before evaluating this table. This can be used to have multiple tables create the same data depending on conditions or to improve performance if tables don't need to be evaluated based on an initial condition(s).
Best Match	If checked, this allows '*' to be used in the values of the data and the best matching row is returned.
Evaluation Order	This indicates the order the tables within the search table group should be evaluated. Starting with 0 and increasing.
Columns	Columns correspond to the 'schema' for each column we're creating for this Custom Reference Data table. • Name: The name of the column in the database. • Display Name: A more readable display name. • Use In Conditions: This represents whether this row will be available for conditions in Policies or Use Case Templates. There is a performance cost to having these checked, so we recommend to uncheck unless they are required. Default value is checked (true). • Type: the type determines what values will be allowed when creating them in control center. • Text: The value is allowed to be any characters. For example, example123!. • Number: The value is allowed to be any whole number. For example, 1234. • Decimal: The value is allowed to be any number (including decimals). For example, 1.234. • True/False: The value needs to be 'true' or 'false'. For example, true. • Date: The value is a date without a time component (May 17th, 2020). • DateTime: The value is a date + time (May 17th, 2020 5:00pm). • Key: This 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 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: This indicates whether this field will be marked required in Control Center. A key is always required.

Parameter	Description
Valid Values	These are the valid values which will be allowed in Control Center (creates a list box). • List of Valid Values: A list of name/display name pairs which 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 Control Center to display an easier to use name. • Valid Values pulled from another Table: This allows initializing the list based on another Custom Reference Data table. The 'name' value will be pulled from another table. There is no way to customize a 'display' name in this manner.
Validation	The validation set here will be checked by Control Center before allowing a row to be added. • Regular Expression: This is the Java regular expression that will be run on the proposed new cell value to validate it as described in http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html. • Regular Expression Description: This is 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).
	• Bind to Subscriber AVP Code: 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 Field: 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

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:

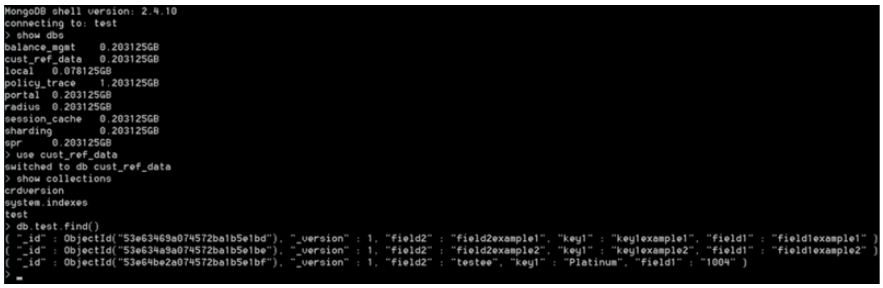
- system.indexes — used by MongoDB. These are indices set on the database.
- crdversion — contains a document indicating the version of all the CRD tables you have defined. The version field increments by 1 every time you make a change or add data to any of your CRD tables.

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 10: CRD Table in Policy Builder



Caching

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, policy decisions utilizing the CRD data won't 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 are different depending on the type of deployment (High Availability or All-in-One):

High Availability (HA): https://<lbvip01>:8443/custrefdata/<tablename>/_<operation>

The examples in the following sections refer to the HA URL.

Query API

Purpose

Returns all rows currently defined in the specified table.

HTTP Operation Type

GET

Example URL

https://<lbvip01>:8443/custrefdata/test/_query

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

Example URL with Filtering

https://<lbvip01>:8443/custrefdata/test/_query?key1=Platinum

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://<lbvip01>:8443/custrefdata/test/_create

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://<lbvip01>:8443/custrefdata/test/_update

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://<lbvip01>:8443/custrefdata/test/_delete

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://<lbvip01>:8443/custrefdata/_checksum

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://<lbvip01>:8443 /custrefdata/_checksum?tableName=<user-provided-table-name>

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://<lbvip01>:8443/custrefdata/<table_name>/_drop

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://<lbvip01>:8443/custrefdata/_export?tableName=<table_name>

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://<lbvip01>:8443 /custrefdata/_export

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.

Export Golden CRD API

Purpose

Exports Golden CRD data to SVN.



Note CPS supports backing up of the existing CRD data and push it to SVN location(s). This backup can be used to restore `cust_ref_data` in case of error scenario(s) after import all.

If there is any kind of error during import all, then CPS stops the process, sets the system in BAD state and blocks CRD APIs execution. CPS also sends error response to the client stating that the system is in BAD state. If system is in BAD state and user restarts QNS/UDC server then CRD cache is built by using golden-crd data. If system BAD state is FALSE, then CRD cache is built using MongoDB.

This enhancement alerts the user about the system state and if the system state is in BAD state, then user has to restore `cust_ref_data` with old and working CRD by using import all API.

Default repository location for golden-crd is: `http://<IP | Hostname>/repos/golden-crd`.

where, <*IP* | *Hostname*> is the IP addresses or hostnames for all the SVN destinations while executing export all proxy API to push existing and working CRD data into SVN.

To know the CRD version from golden-crd's metadata, execute the following command:

```
$ svn cat http://<IP | hostname>/repos/golden-crd/.metadata
```

HTTP Operation Type

GET

Endpoint URL

https://<lbvip01>:8443/custrefdata/_export?goldenCrdHost=<comma separated SVN_HOST>

Example Payload

Response

Success response of API contains the following:

```
Response message: Golden CRD exported successfully

Response Code: 200
```

Failure Response Message:

If Credentials are not correct, the error response is *{"error": "401 Unauthorized"}* and response code is *"401"*.

In case of exceptions, response message is sent to client withresponse code as *"500"*.

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.



Note If you try to import multiple CRD tables during traffic it may have call flow impact. It is recommended to import multiple CRD tables during Maintenance Window (MW).

HTTP Operation Type

POST

Example Endpoint URL

https://<lbvip01>:8443/custrefdata/_import

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.

Import Single File API

Purpose

Imports bulk CRD data by sending any supported file in the API request.

Supports only CSV and XLS file formats.

Note If you try to import single CRD table during traffic it may have call flow impact. It is recommended to import single CRD table during Maintenance Window (MW).

HTTP Operation Type

POST

Example Endpoint URL

https://<lbvip01>:8443/custrefdata/_importsinglefile

Note

1. Error responses are thrown in the following scenarios:
   - When the attached file is of a different format other than CSV and XLS.
   - When an empty file is attached.
   - When the attached file has wrong headers.
   - When the attached file does not have the same file as that of the Policy Builder table name.
   - When the attached file has duplicate records.
   - When no file is attached.
2. Ensure your .xls file does not contain any extra empty and colored header. If the .xls file contains any colored and empty header (header with color but no title), it is considered as a part of the Policy Builder table column. During import file operation, this type of header causes the API to send `Mismatch found between imported csv headers and policy builder table columns` error in response. This is because the empty header is considered as a column from Policy Builder but the Policy Builder table does not contain this empty column.
3. Import Single File API does not support import of SVN CRD table data and displays the following error message:

   Single file import is not allowed for subversion table

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://<lbvip01>:8443/custrefdata/_snapshot?userId=<user_id>&userComments=<user_comments>

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://<lbvip01>:8443/custrefdata/_snapshot

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://<lbvip01>:8443/custrefdata/_revert?snapshotName=<snapshot_name>

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 in the OAM (pcrfclient) VM at the following location:

`/var/log/broadhop/consolidated-qns.log`

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

Policy Builder Publish and CRD Import/Export Automation

Steps for Policy Builder CRD Publish using API:

PB Import to configuration repository > **GET Repository ID (configuration)** > **Reload Repository (configuration)** > **PB Publish to run repository (Using configuration repository ID)** > **Golden CRD Import** > **CRD Import**

SVN Export

Syntax:

```
curl -v -S -k -u <username>:<password> -H Content-Type:application/octet-stream -X
GET http://<server-ip>:<port>/api/repository/actions/export?exportUrl=<svn-url>
-o <cps-extension-file-name>
```

Example:

```
curl -v -S -k -u qns-svn:cisco123 -H Content-Type:application/octet-stream -X
GET http://pcrfclient01:7070/api/repository/actions/export?exportUrl=
http://lbvip02/repos/run/ -o WorkingPB_Backup_run.zip
```

CRD Export

Syntax:

```
curl -v http://lbvip02:8080/custrefdata/_export -o <cps-extension-file-name>
```

Example:

```
curl -v http://lbvip02:8080/custrefdata/_export -o WorkingCRD_Backup.zip
```

SVN Import

Syntax:

```
curl -s -S -k -u <username>:<password> -H Content-Type:application/octet-stream
–data-binary @<cps-extension-file-name> -X POST http://<server-ip>:<port>
/api/repository/actions/import?importUrl=<svn-url>&commitMessage=<message>
```

Example:

```
curl -s -S -k -u qns-svn:cisco123 -H Content-Type:application/octet-stream --trace-ascii
/tmp/dump.txt --data-binary @WorkingPB_Backup_run.zip -X POST
http://pcrfclient01:7070/api/repository/actions/import?importUrl=
http://pcrfclient01/repos/configuration&commitMessage=Importing
```

Create Repositories API

Syntax:

```
curl -v -X POST --progress-bar -H "Content-Type:application/json" --insecure -u
<username>:<password> --data '{"localDirectory":"/var/broadhop/pb/workspace/tmp-<reponame>/","
name":"<reponame>","url":"<svn-url>"}'
"http://lbvip02:7070/api/repository/repositories"
```

Example:

```
curl -v -X POST --progress-bar -H "Content-Type:application/json" --insecure -u
qns-svn:cisco123 --data '{"localDirectory":"/var/broadhop/pb/workspace/tmp-test/",
"name":"test","url":"http://lbvip02/repos/test/"}'
"http://lbvip02:7070/api/repository/repositories"
```

GET Repositories API

Syntax:

```
curl -X GET --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
<username>:<password> "http://pcrfclient01:7070/api/repository/repositories"
```

Example:

```
curl -X GET --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
qns-svn:cisco123 "http://pcrfclient01:7070/api/repository/repositories"
```

Repositiry Reload API

Syntax:

```
curl -v -X PUT --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
<username>:<password> "http://pcrfclient01:7070/api/repository/actions/reload?repository=
<repository_id>"
```

Example:

```
curl -v -X PUT --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
qns-svn:cisco123 "http://pcrfclient01:7070/api/repository/actions/reload?repository=
233bfc09-131e-408a-bdfe-46b70cd72475-49115"
```

PB Publish API

Syntax:

```
curl -v -X PUT --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
<username>:<password> "http://pcrfclient01:7070/api/repository/actions/publish?publishUrl=
http://lbvip02/repos/run&commitMessage=publishing&repository=<repository_ID>"
```

Example:

```
curl -v -X PUT --progress-bar -H "Content-Type:application/octet-stream" --insecure -u
qns-svn:cisco123 "http://pcrfclient01:7070/api/repository/actions/publish?publishUrl=
http://pcrfclient01/repos/run&commitMessage=publishing&repository=
233bfc09-131e-408a-bdfe-46b70cd72475-49115"
```

Golden CRD Export API

Syntax:

```
curl -s -S -k -u <username>:<password> -H Content-Type:application/json -X
GET https://<PB IP>/proxy/custrefdata/_export?goldenCrdHost=lbvip02
```

Example:

```
curl -s -S -k -u qns-svn:cisco123 -H Content-Type:application/json -X
GET https://lbvip01/proxy/custrefdata/_export?goldenCrdHost=lbvip02
```

CRD Import API

Syntax:

```
curl -v -S -k -H Content-Type:application/octet-stream --data-binary
@<name-for-exported-file> -X POST http://lbvip02:8080/custrefdata/
_import?batchOperation=true
```

Example:

```
curl -v -S -k -H Content-Type:application/octet-stream --data-binary
@WorkingCRD_Backup.zip -X POST http://lbvip02:8080/custrefdata/
_import?batchOperation=true
```

Remove Traces of Old Policy Director (LB) VIPs

To remove old lbvips which are not needed, perform the following steps:

Before you begin

Take a backup of the following two files for future reference.

`/var/qps/config/deploy/csv/VLANs.csv`

`/var/qps/config/deploy/csv/AdditionalHosts.csv`

Procedure

Step 1 From Cluster Manager:

a) Remove the entries for the lbvips which are not required from `VLANs.csv` and `AdditionalHosts.csv` file.

- Sample `VLANs.csv` entry to be removed.

  ```
  Gy,VM Network,255.255.255.0,NA,lbvip05,,eth3
  ```
- Sample `AdditionalHosts.csv` entry to be removed.

  ```
  lbvip05,lbvip05,123.45.67.89
  ```

b) Execute the following commands to import the updated files to all the VMs:

```
/var/qps/install/current/scripts/import/import_deploy.sh
/var/qps/install/current/scripts/build_all.sh
/var/qps/install/current/scripts/upgrade/reinit.sh
```

Step 2 From Policy Director (LB):

a) Execute the following command to delete th extra VIP:

```
pcs resource delete RESOURCE_ID
```

where, is the *RESOURCE_ID* of the lbvip which you want to remove.

Example: `pcs resource delete lbvip05`