Working with CPS Utilities

Policy Tracing and Execution Analyzer

Cisco Policy Server comes with a set of utilities to actively monitor and trace policy execution. These utilities interact with the core policy server and the mongo database to trigger and store traces for specific conditions.

Architecture

The policy tracing and execution analyzer is 3-tier architecture:

- Tier 1 — command line utilities to manage the policy trace generation and extract policy traces.
- Tier 2 — policy server creation of policy traces using triggers defined in Tier 1.
- Tier 3 — storage of the policy traces in a MongoDB.

Administering Policy Traces

All commands are located on the Control Center virtual machine within `/var/qps/bin/control` directory. There are two main scripts which can be used for tracing: `trace_ids.sh` and `trace.sh`.

- The `trace_ids.sh` script maintains all rules for activating and deactivating traces within the system.
- The `trace.sh` script allows for the real time or historical retrieval of traces.

Before running `trace_ids.sh` and `trace.sh`, confirm which database you are using for traces. For more information, refer to Policy Trace Database, on page 5. If no database has been configured, then by default the scripts connects to primary database member of SPR-SET1.

Managing Trace Rules using trace_ids.sh

Running `trace_ids.sh` with *-h* arguments produces a help text describing the capabilities of the script.

```
/var/qps/bin/control/trace_ids.sh -h
```

Usage:

```
/var/qps/bin/control/trace_ids.sh -i <specific id> -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace_ids.sh -i <specific id> -c <command code> -d
sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace_ids.sh -r <specific id> -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace_ids.sh -x -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace_ids.sh -l -d sessionmgr01:27719/policy_trace
```

Note By default , if *-d* option is not provided then the script connects to primary database member of SPR-SET1. If you are not using the SPR database, you need to find out the which database you are using. To find out which database you are using, refer to Policy Trace Database, on page 5. Make sure to update the commands mentioned in Step 1, on page 2 to Step 5, on page 2 accordingly.

This script starts a selective trace and outputs it to standard out.

Step 1 Specific audit ID tracing:

```
/var/qps/bin/control/trace_ids.sh -i <specific id>
```

Step 2 Specific audit ID tracing with specific command codes:

```
/var/qps/bin/control/trace_ids.sh -i <specific id> -c <command code>
```

<command code> is case sensitive and is used to trace specific control command message. For example, Gx_CCR-I, Sh_UDR and so on. You can add multiple command code separated by comma.

Step 3 Remove trace for specific audit ID:

```
/var/qps/bin/control/trace_ids.sh -r <specific id>
```

Step 4 Remove trace for all IDs:

```
/var/qps/bin/control/trace_ids.sh -x
```

Step 5 List all IDs under trace:

```
/var/qps/bin/control/trace_ids.sh -l
```

Adding a specific audit ID for tracing requires running the command with the *-i* argument and passing in a specific ID. The policy server matches the incoming session with the ID provided and compares this against the following network session attributes:

- Credential ID
- Framed IPv6 Prefix
- IMSI
- MAC Address
- MSISDN

- User ID

If an exact match is found then the transaction are traced. Spaces and special characters are not supported in the audit ids.

- Removing a specific audit id from active tracing requires specifying the *-r* argument with id to remove.
- Removing all ids requires sending in the *-x* argument and this will remove all ids from the database.
- Listing all ids requires sending in the *-l* argument.

Usage:

Usage with SPR-SET as database:

```
#./trace_ids.sh -l
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27720/policy_trace
112345
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27720/policy_trace
null
```

Usage with *-d* option:

```
#./trace_ids.sh -l -d sessionmgr01:27717/policy_trace
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27717/policy_trace
874838
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27717/policy_trace
null
```

Situations where traces are generated automatically

The following criteria cause the system to generate a trace regardless of whether the id is present in the trace database or not:

- If there is an AVP with the code: audit_id, audit-id, auditid. In this case, the traces are stored in the database with the value of the AVP.
- If there is a subscriber attribute (USuM AVP) with a code of audit-policy and a value of “true”. In this case, the traces are stored using the credentials stored for the subscriber.
- If an error is triggered internally.

Note An error is defined as an internal processing error (e.g. database failure or other failure) and is not a failure message code.

Managing Trace Results using trace.sh

Running `trace.sh` with *-h* arguments produce a help text describing the capabilities of the script:

```
/var/qps/bin/control/trace.sh -h
```

Usage:

```
 /var/qps/bin/control/trace.sh -i <specific id> -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace.sh -x <specific id> -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace.sh -a -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace.sh -e -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace.sh -f -d sessionmgr01:27719/policy_trace
    /var/qps/bin/control/trace.sh -h
```

Note By default , if *-d* option is not provided then the script connects to primary database member of SPR-SET1. If you are not using the SPR database, you need to find out the which database you are using. To find out which database you are using, refer to Policy Trace Database, on page 5. Make sure to update the commands mentioned in Step 1, on page 4 to Step 4, on page 4 accordingly.

This script starts a selective trace and outputs it to standard out.

Step 1 Specific audit ID tracing:

```
/var/qps/bin/control/trace.sh -i <specific id>
```

Specifying the *-i* argument for a specific ID causes a real time policy trace to be generated while the script is running. Users can redirect this to a specific output file using standard Linux commands.

Step 2 Dump all traces for specific audit ID:

```
/var/qps/bin/control/trace.sh -x <specific id>
```

Specifying the *-x* argument with a specific ID, dumps all historical traces for a given ID. Users can redirect this to a specific output file using standard Linux commands.

Step 3 Trace all:

```
/var/qps/bin/control/trace.sh -a
```

Specifying the *-a* argument causes all traces to output in real time policy trace while the script is running. Users can redirect this to a specific output file using standard Linux commands.

Step 4 Trace all errors:

```
/var/qps/bin/control/trace.sh -e
```

Specifying the *-e* argument causes all traces triggered by an error to output in real time policy trace while the script is running. Users can redirect this to a specific output file using standard Linux commands.

Step 5 Flush out the trace collection:

```
/var/qps/bin/control/trace.sh -f -d <DB>
```

Policy Trace Database

The default location of the policy trace database is the administrative database and can be optionally specified in the trace database fields. These fields are defined at the cluster level in the system configurations.

Note Make sure to run all trace utility scripts from `/var/qps/bin/control` directory only.

Configure Traces Database in Policy Builder

Step 1 Log in to the Policy Builder.

Step 2 From left pane, open up the *name of your system* and select the required cluster.

Step 3 From right pane, select the check box for **Trace Database**.

The following table provides the parameter descriptions under **Trace Database** check box:

Table 1: Trace Database Parameters

Parameter	Description
Primary Database IP Address	The IP address of the sessionmgr node that holds trace information which allows for debugging of specific sessions and subscribers based on unique primary keys.
Secondary Database IP Address	The IP address of the database that provides fail over support for the primary database. This is the mirror of the database specified in the Primary IP Address field. Use this only for replication or replica pairs architecture. This field is present but deprecated to maintain downward compatibility.
Database Port	Port number of the database for Session data. Default value is 27717.

Network Cutter Utility

CPS supports a new network cutter utility, which keeps monitoring Policy Server (QNS) VMs failures. When any of the Policy Server VMs are down, utility cuts those unnecessary connections to avoid sending traffic to Policy Server VMs that are down, and this also results in avoiding timeouts.

This utility is started by `monit` on Policy Director (lb) VMs and keeps monitoring policy server VMs failures.

Utility stores log on `/var/log/broadhop/network-cutter.log` file.

You can verify the status of network cutter utility on lb01/02 VMs using `monit summary` and `network-cutter` status command:

```
monit summary | grep cutter
Process 'cutter' Running
```

```
service network-cutter status
network-cutter (pid 3735) is running
```

You can verify if network cutter utility has been started using `ps -ef | grep cutter` command:

```
ps -ef | grep cutter
root 6496 1 0 Feb18 ? 00:16:22 /usr/java/default/bin/java -jar
/var/broadhop/images/network-cutter.jar
```

Note In CPS, CPU% consumption is high for the Policy Director (lb) that has lbvip configured than the Policy Director (lb) where lbvip is not present. This is due to netstat commands running for the network-cutter feature.

Policy Builder Configuration Reporter

The Configuration-Reporter utility processes CPS Policy Builder configuration and report any missing cross-reference files and stale files. An option has also been provided to remove the stale files and missing cross-references in the XMI files from the configuration data in the utility.

Important This utility can be used before or after installation to check if customers have all the configuration files needed.

This reporting utility address the following concerns:

- Reports if there are any missing PB configuration files (.xmi files) and a summary of what those files are.
- Reports if there are any stale files and a summary of the same.

  Stale files are Service Option files whose corresponding Use Case Template files are missing.
- It also shows the missing configuration files on a per-file basis, showing the files that are referencing the missing files.
- Additionally, the customer can see all the different configuration objects and their quantity to see the variety of configurations they are using.
- Using -r option, utility creates a new archive file with cleaned XMI files (removes the stale files and missing cross-references from XMI files from the original configuration data).

To run the utility, perform the following steps:

1. Mount ISO on Cluster Manager if you unmounted the ISO after completing the CPS installation or upgrade.
2. Extract the release train into the temp directory:

   ```
   cd /tmp
   ```

   ```
   tar -zxvf /mnt/iso/app/install/release-train-xxx.tar.gz
   ```

   where, `release-train-xxx.tar.gz` is the release train version.
3. Go into Configuration-Reporter directory which is present inside utility directory of extracted utility.

   ```
   cd release-train-xxx/Utility/Configuration-Reporter
   ```

4. Execute jar using the following command:

   ```
   java -jar configuration-reporter.jar <pb-configuration-xmi-files-in-archive-form> [-r]
   ```

   where,

   - *<pb-configuration-xmi-files-in-archive-form>* is the name of the configuration file.
   - `[-r]` is an optional parameter and if specified will remove all the references of missing files from XMI files and stale files in the archive file and outputs the corrected archive as `filename_cleaned.zip|cps` (output file will have same extension as input file) on the same path where command runs.

CRD Generator Conversion Tool

CPS provides a CRD conversion tool which converts existing Balance and Quota templates PB configuration data to CRD Data. You can provide XMI files to the tool in the following ways:

- Use Import/Export tool to export CPS configuration as an archive file (.cps extension archive) and provide the same to the tool.
- Archive set of XMI files to .zip extension archive file.
- Provide directory path where XMI files are present as an input to the tool.

Important The conversion tool is used to convert all Balance and Quota template configuration data to CRD data. This helps to reduce the number of XMI files in the system and improves the performance.

Prerequisites:

The feature `com.broadhop.balance.crdbalance.feature` must be enabled so that CRD tables for Balance and Quota Template details are displayed in Policy Builder (as readonly) and Control Center. These CRD tables need to be present for importing the Balance and Quota CRD data which will be converted using the tool and Balance and Quota Templates XMIs present in Policy Builder.

To enable `com.broadhop.balance.crdbalance.feature`, add the feature in `/var/qps/current_config/etc/broadhop/pb/features` and `/var/qps/current_config/etc/broadhop/pcrf/features` files. For more information, refer to *Customize Features in the Deployment* section in *CPS Installation Guide for VMware*.

To run the utility, perform the following steps:

1. Mount ISO on Cluster Manager.
2. Extract release train into temp directory:

   ```
   tar -zxvf /mnt/iso/app/install/xxx.tar.gz /tmp/
   ```

3. Go to `CRD_generator_Utility` directory which is inside the utility directory of the extracted release train:

   ```
   cd /tmp/release-train-xxx/Utility/CRD_generator_Utility
   ```

4. Execute jar using the following command:

```
java –jar com.broadhop.customreferencedata.generator-<svn-revision-number>-full.jar [-a
 <archive-file> | -d <directory>]
```

The following table describes the various command line options:

Command Line Options	Description
-a	Option for passing zip archive file which contains XMI files.
-d	Option for passing directory path where XMI files are present.
-e	Generates .exportCrdInfo file with specified exportCRDversion. Valid Values 1 and 2 are described as follows: **a.** 1 : Data-type validations will happen only during import of generated archive into CPS. **b.** 2 : All validations will happen during import of generated archive into CPS and is the default value.
-h	Prints help.
-o	Object type for which CRD conversion needs to be performed. Default value is AccountBalanceTemplate.
-r	Removes duplicate CRD data. Valid Values 0 and 1 are described as follows: **a.** 0 : De-duplication is disabled by default which means duplicate data will be part of generated CRD data files. **b.** 1 : Keeps the first record is retained and skips the rest. De-duplication is enabled which means duplicate data will be not be part of generated CRD data.
-v	Validates CRD data against the schema (required field constraint validation). Default value is true which means validation of schema is enabled and CRD data record with missing required field value will not be part of generated CRD data files.
-xls	Generates XLS format files for CRD data. Default option is CSV format CRD data files.

The tool generates a ".crd" extension archive file containing ".exportCrdInfo" file and CRD tables data in CSV/XLS format which can be used by Import/Import All CRD functionality in CPS to import the CRD data into the system.

5. To view or edit the csv files, perform the following optional steps:
   a. View-only using Excel : In Excel, you can only view the csv files present in generated ".crd" archive where non-ASCII characters are present in it. In order to view non-ASCII characters which might be present in CRD Data, perform the following steps:
      1. Open a blank Excel file.
      2. Go to Menu option **Data** > **From Text File** and import the CRD table CSV file in which non-ASCII is present.
      3. A "Text Import Wizard" is displayed. Perform the following steps:
         a. Select Unicode (UTF-8) in File origin drop-down.
         b. Check **Comma** as delimiters.
         c. Do not perform any changes and click **Finish**.
         d. Click **Ok**.
      4. All non-ASCII characters are displayed correctly.

Note It is not recommended to edit generated CRD Table csv files containing non-ASCII characters in excel view.

   b. View and edit using other editors (vi editor): You can view and edit csv files present in ".crd" archive file using editors such as vi editor even if the CRD data contains non-ASCII characters.

6. The generated ".crd" extension archive file needs to be imported as CRD Data into CPS which can be performed using the following options:
   - Use "_import" CRD API to import CRD data in CSV format.
   - Use "Import All" option in Control Center to import CRD data in CSV format.
   - Use "Import" option in Control Center to import CRD data in XLS format. This option enables you to import single XLS CRD data at a time.

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).

Policy Builder Configuration Converter Conversion Tool

CPS provides a conversion tool to convert the balance references in the existing service configuration to CRD data string value to adopt the CRD table driven configuration solution. The tool can perform the following:

- Convert Account Balance template references present in existing Customer's PB Service configuration to CRD Data "Dynamic Reference Data Key" string value.
- An "-r" option is provided to clean up the following converted referenced data:
  - References to Account Balance template in Service Options, Use Case Templates and Use Case Options is removed in the output archive file configuration data.
  - Account Balance template and all Quota templates present in the original PB configuration data will not be part of output archive file.

Important This conversion tool is used to convert balance references in existing configuration data and clean Balance and Quota templates as part of result archive file. This helps in reducing the number of XMI files in configuration data.

To run the utility, perform the following steps:

1. Mount ISO on Cluster Manager.
2. Extract release train into temp directory:

   ```
   tar -zxvf /mnt/iso/app/install/xxx.tar.gz /tmp/
   ```

3. Go to `PB-Configuration-Converter_Utility` directory which is inside the utility directory of the extracted release train:

   ```
   cd /tmp/release-train-xxx/Utility/PB-Configuration-Converter_Utility
   ```

4. Execute jar using the following command:

   ```
   java –jar pb-configuration-converter-<svn-revision-number>-full.jar [-a <archive-file>
   | -d <directory> ] [-r]
   ```

   a. You have the option to provide the XMI files input as an archive file or directory path in which all Policy Builder created XMI files are present. Select any one of the following mandatory options to run the command:
      - -a : Option for passing Archive file (.zip or .cps extension archive file).
      - -d : Option for passing directory path containing XMI files to process.

   b. You can use "-r" option to perform cleanup operation for reducing the XMI files as follows:
      - Removes Account Balance template references from Service Option XMIs once the update of “Dynamic Reference Data Key” is performed.
      - Removes Account Balance template references from Use Case Template and Use Case Option XMI files.
      - Removes Account Balance template , One Time Quota template, Recurring Quota template and Rollover Quota template XMI files from PB configuration data in resulting archive file.

The tool generates an archive file named as “<input-file-name>_updated.<input-file-extension>” if it is an archive file input or "<input-directory-name>_updated.zip" if it is a directory file input.. It contains all the XMI files in the input file along with updated Service Option XMI files with a new field “dynamicRefDataKey” if there are references to Account Balance template object type.

Note The output archive file might not contain ".exportInfo" and ".exportRepositoryInfo" files as the tool only works on conversion of Service configuration balance reference data present in user input and copies all other input files in the output archive.

Modifying Audit Rule File

You need to modify the `audit.rules` file as described in the following section.

Step 1 Add new rules to `etc/audit/rules.d/audit.rules` file.

Sample Audit Rule File

```
cat audit.rules:
-D
-b 8192
-w /etc/group -p wa -k identity
-w /etc/passwd -p wa -k identity
-w /etc/shadow -p wa -k identity
```

Step 2 Restart the auditd service by executing the following command:

```
service auditd restart
```

After restart, the audit files are loaded in the `audit.rules` file `/etc/audit` location.

Step 3 Execute the following command to view all the implemented audit rules:

```
auditctl -l
```

Step 4 Execute the following command to verify if the audit rule file is correct and error-free:

```
auditctl -R /etc/audit/audit.rules
```

If the audit.rule file has any error, it is highlighted in the output.

Step 5 Monitoring of logging for critical file is done in the following location on cluster manager:

```
/var/log/audit/ audit.log
```

Step 6 Execute the following command to validate particular audit logs:

```
ausearch -i -k <key to moniter>
e.g. ausearch -i -k "identity"
---
time->Fri Aug 24 11:29:19 2018
type=PROCTITLE
msg=audit(1535110159.513:18131):proctitle=7573657264656C00736F6D61
type=PATH msg=audit(1535110159.513:18131): item=0 name="/etc/passwd"
inode=2886645 dev=08:02 mode=0100644 ouid=0 ogid=0 rdev=00:00 objtype=NORMAL
type=CWD msg=audit(1535110159.513:18131):  cwd="/root"
type=SYSCALL msg=audit(1535110159.513:18131): arch=c000003e syscall=2 success=yes
exit=5 a0=5595b6cd69e0 a1=20902 a2=0 a3=8 items=1 ppid=25552 pid=26946 auid=0 uid=0
gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=984 comm="userdel"
exe="/usr/sbin/userdel" key="identity"
```

```
time->Fri Aug 24 11:45:23 2018
----
type=CONFIG_CHANGE msg=audit(1535111123.222:18175): auid=4294967295 ses=4294967295
op=remove_rule key="identity" list=4 res=1
```

Note You can add the following rules to the audit file to montior each file:

```
-w /var/run/utmp -p wa -k session

-w /var/log/wtmp -p wa -k session

-w /var/log/btmp -p wa -k session

-w /etc/login.defs -p wa -k password

-w /etc/group -p wa -k identity

-w /etc/passwd -p wa -k identity

-w /etc/shadow -p wa -k identity

-w /etc/sudoers -p wa -k identity

-w /etc/fstab -p wa -k quota_rate_limit

-w /etc/iptables -p wa -k network
```

`Identity`, `quota_rate_limit`, and `network` are keywords used to search the respective log.

Support for CPS Auto Healing in Case of Endpoint Heart Beat Failures

CPS supports an auto healing and auto correction in case of endpoint heart beat failures between the Policy Director (LB) and Policy Server (QNS) VMs IPC connections. When there is a heart beat failure, application generates the down counter for the same.

To support the auto correction of the qns process, two python scripts are utilized which runs through monit to monitor failure counters continuously. Depending on the heal option available, you need to enable or disable it to take necessary actions.

There are two scripts which monitor the counts and take the necessary actions:

- Auto heal client script which runs on Policy Server (QNS) and Policy Director (LB) VMs via monit collect the IPC connection down counters and if the IPC connection continue fails then the VM details are stored in the same VM.
- Auto heal server script which runs on pcrfclient (OAM) VMs via monit collects the details of the failed VM that are generated by the auto heal client script and sends the alarm for the same.

If the script is running with auto heal enabled, then the script sends an alarm and recovers the system by restarting the qns process. Before restarting the process, the script ensures that there is no other issues (such as, network connectivity or manual stop of qns process) on the VM. If present then the script skips the recover process for that qns process.

If the script is running with auto heal disabled, then it skips to recover the qns process but it sends an alarm.

The following section describes two scripts:

auto_heal_server.py

Server script runs on pcrfclient VMs.

```
usage: auto_heal_server.py [-h] [--maxRestartCount MAXRESTARTCOUNT]
                           [--restartPeriod RESTARTPERIOD]
                           [--enableMode ENABLEMODE]
                           [--maxLogfileSize MAXLOGFILESIZE]
                           [--maxLogfilesCount MAXLOGFILESCOUNT]
                           [--waittime WAITTIME]
CPS Auto Healing
optional arguments:
  -h, --help            show this help message and exit
required named arguments:
  --maxRestartCount MAXRESTARTCOUNT, -m MAXRESTARTCOUNT
                        Maximum number of allowed restarts of QNS process
                        within a period
  --restartPeriod RESTARTPERIOD, -p RESTARTPERIOD
                        Period in seconds in which script allowed to be wait
                        next restart of qns process in case process not came
                        up.
  --enableMode ENABLEMODE, -e ENABLEMODE
                        Enabling the auto healing function, when flag true it
                        will restart QNS process
  --maxLogfileSize MAXLOGFILESIZE, -ls MAXLOGFILESIZE
                        Maximum size of the log file in bytes
  --maxLogfilesCount MAXLOGFILESCOUNT, -lc MAXLOGFILESCOUNT
                        Maximum number of logs to be keep in the log path
  --waittime WAITTIME, -w WAITTIME
                        Wait time in sec to get the process status from QNS
                        and LB VMs from client, if client still running.
```

auto_heal_client.py

Client script runs on all LB and QNS VMs.

```
usage: auto_heal_client.py [-h] [--intervalSleep INTERVALSLEEP]
                           [--maxLogfileSize MAXLOGFILESIZE]
                           [--maxLogfilesCount MAXLOGFILESCOUNT]
                           [--waittime WAITTIME]

CPS Auto Healing

optional arguments:
  -h, --help            show this help message and exit

required named arguments:
  --intervalSleep INTERVALSLEEP, -i INTERVALSLEEP
                        Failure counter fetch interval sleep time in sec
  --maxLogfileSize MAXLOGFILESIZE, -ls MAXLOGFILESIZE
                        Maximum size of the log file in bytes
  --maxLogfilesCount MAXLOGFILESCOUNT, -lc MAXLOGFILESCOUNT
                        Maximum number of logs to be keep in the log path
  --waittime WAITTIME, -w WAITTIME
                        Wait time in sec to get the process status from QNS
                        and LB VMs from client, if client still running.
```

Log Collector

The `logCollector.sh` script provides the following operations:

- Provides options to enable and disable the log levels for specific components, class, and interfaces.
- Provides the timer function in the script to collect specific logs for certain log level. Once the timer expires, the logs revert to the default log level.
- Provides an option to store the logs at a specified path or default path.
- Provides option to collect the application logs and MongoDB logs.

Script location: `/var/qps/bin/support/logCollector.sh`

For more information on the `logCollector.sh` script, refer to logcollector.sh.