



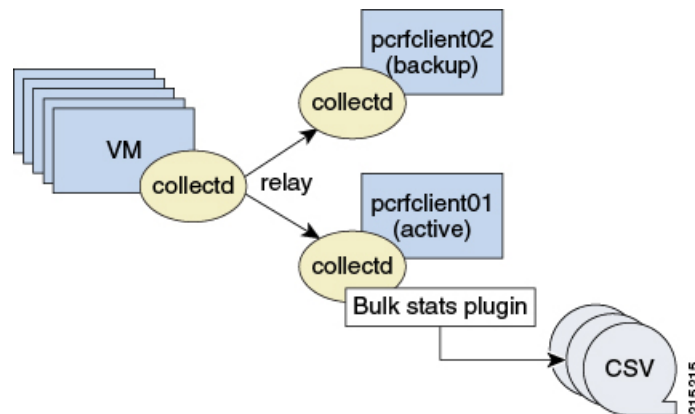
## CPS Statistics

- [Bulk Statistics Overview](#), on page 1
- [CPS Statistics](#), on page 2
- [Bulk Statistics Collection](#), on page 5
- [CPS KPI Monitoring](#), on page 8
- [Example CPS Statistics](#), on page 32

## Bulk Statistics Overview

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

**Figure 1: CPS Bulk Statistic Collection Architecture**



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

`Collectd` within CPS is deployed with nodes relaying data using the `collectd` network plug-in (<https://collectd.org/wiki/index.php/Plugin:Network>) to the centralized collection nodes on the `pcrfclient01` and `pcrfclient02` virtual machines. The centralized collector writes the collected data to output CSV files.



**Note** pcrfclient01 and pcrfclient02 collect bulk statistics independently. As a result, it is normal to have slight differences between the two files. For example, pcrfclient01 generates a file at time  $t$  and pcrfclient02 generates a file at time  $t \pm$  the clock drift between the two machines.

As a best practice, always use the bulk statistics collected from pcrfclient01. pcrfclient02 can be used as a backup if pcrfclient01 fails.

If pcrfclient01 becomes unavailable, statistics is still gathered on pcrfclient02. Statistics data is not synchronized between pcrfclient01 and pcrfclient02, so a gap exists in the collected statistics while pcrfclient01 is down.



**Note** Statistics value in csv files is displayed in E notation format depending on value and data source type. For example, for Gauge type of data source, statistics value is converted to E notation if value is greater than  $10^7$ .

## Grafana

## CPS Statistics

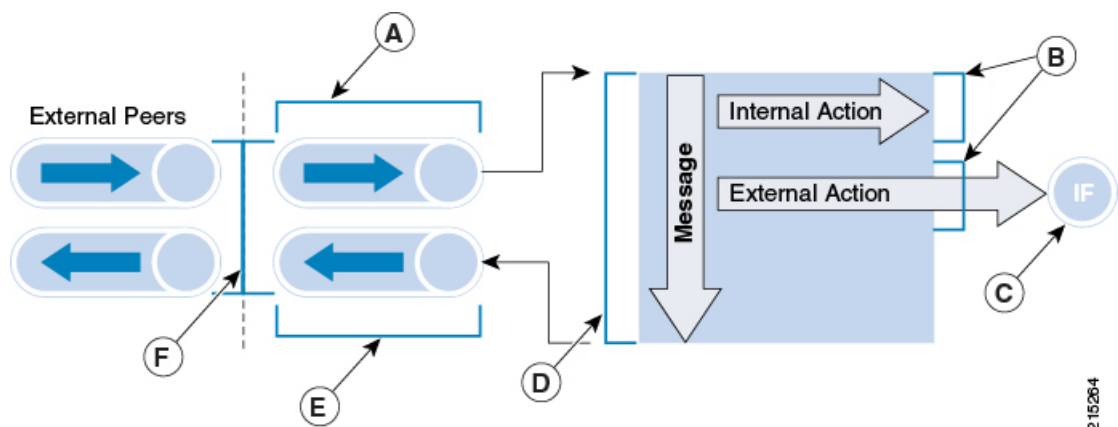
The list of statistics available in CPS is consolidated in an Excel spreadsheet. After CPS is installed, this spreadsheet can be found in the following location on the Cluster Manager VM:

```
/var/qps/install/current/scripts/documents/QPS_statistics.xlsx
```

## Overview

The following diagram represents the various statistic gathering points for incoming and outgoing messages.

**Figure 2: Various Statistic Gathering Points for Incoming and Outgoing Messages**



215264

**Table 1: Measurement Legend**

| Legend  | Description                         |
|---|-------------------------------------|
| A   | Inbound queue counts and times*     |
| B   | Policy action counts and times      |
| C   | Interface specific counts and times |
| D   | Policy message counts and times     |
| E   | Outbound queue counts and times*    |
| F   | Round trip counts and times*        |
| where, * – statistics only apply to Diameter messages |                                     |

A brief description of each statistic gathering points is given below:

- Upon receipt of a message on the Policy Director (lb) node, the message is registered as received and forwarded to a middle tier processing node.
- This middle tier processing node tracks the inbound message counts and time spent within the inbound processing queue. If a message is discarded due to SLA violation, then counters are incremented at this point. This occurs at point A within the diagram.
- Upon arrival within the policy engine all messages are counted and timers are started to measure the duration of processing.
- Any internal or external actions are tracked at this point and the round trip time is measured from the policy engine invocation of the action and success or failure of the action. This occurs at point B within the diagram.
- For external actions (for example, LDAP), interface specific statistics maybe captured. This occurs at point C in the diagram and is gathered from the Policy Director nodes.
- Upon completion of the message in the policy engine, the total elapsed time is measured and whether success or failure occurred in processing.



**Note** A message is considered a success even if the policy returns an error (such as 5002). These application errors are tracked at point D within the diagram.

- Outbound messages are tracked from the policy engine to the Policy Directors at point E within the diagram.
- Upon receipt of outbound messages, the Policy Directors tracks either end to end completion time for inbound requests OR starts a timer and counts outbound requests. This occurs at point F within the diagram.

## CPS Statistic Types

This section describes various forms of statistics generated by CPS.

### Diameter Statistics

In Diameter statistics, Monitoring Areas are defined on the basis of Queues maintained in it. Diameter statistics can also be defined based on whether the statistic is related to a counter or gauge or derived or absolute.

- Counter: Counter type represents a non-negative integer which monotonically increases until it reaches a maximum value of  $2^{32}-1$  (4294967295 decimal), when it resets and starts increasing again from zero.

Counters have no defined “initial” value, and thus, a single value of a Counter has (in general) no information content. You must take a delta of multiple readings to understand anything.

- Gauge: Gauge type represents a non-negative number, which can increase or decrease, but can never fall below a minimum value. The minimum value cannot be smaller than 0.
- Derived: It is intended to store the derivative of the line going from the last to the current value of the data source. Such data sources are very common with events that can be counted. Internally, derive works exactly like COUNTER but without overflow checks. So if your counter does not reset at 32 or 64 bit you might want to use DERIVE and combine it with a MIN value of 0.
- Absolute: It is intended for counters which get reset upon reading. In effect, the type is very similar to GAUGE except that the value is an (unsigned) integer and is divided by the time since the last reading. This is used for fast counters which tend to overflow. So instead of reading them normally you reset them after every read to make sure you have a maximum time available before the next overflow. Another usage is for things you count like number of messages since the last update.

### LDAP Statistics

CPS tracks LDAP statistics for general LDAP actions, LDAP query counters, LDAP connection counters, as well as message counters.

Categories:

- Action
- Messages

### System Statistics

System statistics are defined based on six categories:

- CPU
- File System Usage
- Disk Performance
- Interface
- CPU Load
- Memory

## Engine Statistics

Engine statistics are defined based on three categories:

- Session Count
- Session Operation
- Internal messages

## Error Statistics Definitions

About error statistics, here are the definitions of each error suffix:

**Table 2: Error Statistics Definitions**

| Error Statistics       | Description   |
|------------------------|---|
| node1.messages.*.error | Failure processing a message  |
| e2e*_qns_stat.error    | Count of occurrence for given Diameter result code                  |
| pe-submit-error        | Error submitting to policy engine                                   |
| _bypass                | Message not sent to policy engine due to successful response (2001) |
| _drop                  | Message dropped due to SLA violation                                |
| rate-limit             | Message dropped due to rate limiting violation                      |



**Note** The Diameter E2E statistics with the suffix “error” always have a value of 0 (zero) unless they have “\_late” in the statistic name.

## Bulk Statistics Collection

By default, CPS outputs a bulk statistics CSV file to the /var/broadhop/stats/ directory on the perfcient01 and perfcient02 VMs in five minute intervals.

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

These CSV files include all statistics collected from all VMs during the 5 minute interval.



**Note** If a statistic is generated by the system multiple times within the 5 minute interval, only the last measured statistic is collected in the CSV file.

The following list is a sample of the file names created in the /var/broadhop/stats/ directory on the perfcient01 VM.

```
[root@pcrfclient01 stats]# pwd
/var/broadhop/stats
[root@pcrfclient01 stats]# ls
bulk-pcrfclient01-201510131350.csv
bulk-pcrfclient01-201510131355.csv
bulk-pcrfclient01-201510131400.csv
bulk-pcrfclient01-201510131405.csv
bulk-pcrfclient01-201510131410.csv
bulk-pcrfclient01-201510131415.csv
bulk-pcrfclient01-201510131420.csv
bulk-pcrfclient01-201510131425.csv
bulk-pcrfclient01-201510131430.csv
bulk-pcrfclient01-201510131435.csv
bulk-pcrfclient01-201510131440.csv
bulk-pcrfclient01-201510131445.csv
bulk-pcrfclient01-201510131450.csv
bulk-pcrfclient01-201510131455.csv
bulk-pcrfclient01-201510131500.csv
bulk-pcrfclient01-201510131505.csv
bulk-pcrfclient01-201510131510.csv
bulk-pcrfclient01-201510131515.csv
bulk-pcrfclient01-201510131520.csv
bulk-pcrfclient01-201510131525.csv
bulk-pcrfclient01-201510131530.csv
bulk-pcrfclient01-201510131535.csv
bulk-pcrfclient01-201510131540.csv
bulk-pcrfclient01-201510131545.csv
bulk-pcrfclient01-201510131550.csv
bulk-pcrfclient01-201510131555.csv
bulk-pcrfclient01-201510131600.csv
bulk-pcrfclient01-201510131605.csv
bulk-pcrfclient01-201510131610.csv
bulk-pcrfclient01-201510131615.csv
bulk-pcrfclient01-201510131620.csv
bulk-pcrfclient01-201510131625.csv
bulk-pcrfclient01-201510131630.csv
```

## Retention of CSV Files

CPS retains each bulk statistic CSV file on the pcrfclient01/02 VM for 2 days, after which the file is automatically removed. If you need to preserve these CSV files, you must back up or move them to an alternate system.

## Configuring Logback.xml

Configuration of the CPS application statistics is controlled in the `/etc/collectd.d/logback.xml` file.

Refer to <http://logback.qos.ch/manual/appenders.html> for more information about the configuration of the `logback.xml` file.

Collectd is configured in the following files:

- `/etc/collectd.conf`
- `/etc/collectd.d/jmxplugin.conf`
- `/etc/collectd.d/exec.conf`

## Restarting the Collectd Service

After making any configuration changes to logback.xml, restart the collectd service:

```
monit restart collectd
```

## Adding Realm Names to Diameter Statistics

By default, the Diameter statistics that are generated do not include the realm names. To include realms in the statistics collected, add the following line in the qns.conf file (comma separated auth-appl-id).

```
-Ddiameter.appid.realm.stats=Auth-Appl-Id-1,Auth-Appl-Id-2,... Auth-Appl-Id-n
```

where each Auth-Appl-Id refers to the specific protocol's Auth-Application-Id for which realms are needed in the statistics.

For example, to add Gx, Gy, Rx and Sy realms to the statistic names, use the following Auth-Appl-Ids:

```
-Ddiameter.appid.realm.stats=16777238,16777235,16777236,9
```

where

- Gx Auth-Application-ID = 16777238
- Rx Auth-Application-ID = 16777236
- Gy Auth-Application-ID = 4
- Sy Auth-Application-ID = 7




---

**Note** Adding a realm will increase the number of statistics generated/collected. Add realms only when necessary.

---

As an example, statistic names with and without the realms are shown below for reference for the following statistic:

```
e2e_<domain>_[realm_] [alias_] <message id>
```

### Counter name with Realm (with qns.conf file modification):

```
C,lb02,node2.messages.e2e_PHONE_sy-ac.cisco.com_AC_Syp_AAR_2001.qns_stat.success,528
```

```
C,lb02,node2.messages.e2e_PHONE_sy-bm.cisco.com_BM_Syp_AAR_2001.qns_stat.success,1221
```

### Counter name without Realm (without qns.conf file modification):

```
C,lb01,node2.messages.e2e_PHONE_AC_Syp_AAR_2001.qns_stat.success,1495
```

```
C,lb01,node2.messages.e2e_PHONE_BM_Syp_AAR_2001.qns_stat.success,4
```

Each statistic field has a fixed maximum length of 63 characters. Based on the current syntax, the length of the realm should not exceed 16 characters, otherwise it will lead to truncation of the counter name.

## CPS KPI Monitoring

This section provides a list of Key Performance Indicators (KPIs), useful for tracking the overall health of CPS.

The complete list of CPS statistics is available in a spreadsheet format in the following location on the Cluster Manager VM:

```
/var/qps/install/current/scripts/documents/QPS_statistics.xlsx
```

The KPIs highlighted in the following sections are also included on the **Stats Recommended to Monitor** tab in the `QPS_statistics.xlsx` spreadsheet.

## System Health Monitoring KPIs

The following table lists the KPIs and thresholds to track the overall performance of the CPS deployment, including information about the underlying hardware.

**Table 3: System Health Monitoring KPIs**

| Name/Description  | Statistics/Formula                        | Warning Threshold  | Major Threshold   |
|---|---|--|---|
| <b>CPU Utilization</b><br>CPU is a critical system resource. When the demand increases and CPU utilization exceeds 80% utilization, the efficiency of the CPU is reduced. When CPU utilization exceeds 80%, the application processing time will increase, message response will increase, and drops and timeouts will be seen. | <code>100 - cpu.&lt;cpuid&gt;.idle</code> | > 60% utilization over 60 second period<br><br>(assuming that idle is less than 40%) | > 80% utilization over 60 second period<br><br>(assuming idle is less than 20%) |
| <b>CPU Steal</b><br>If multiple VMs on the same hypervisor and same hardware have concurrent CPU demands, the hypervisor will “steal” CPU from one VM to satisfy another VM CPU needs. If the CPU Steal statistic is non-zero, there is not enough CPU allocated for the VMs.   | <code>cpu.&lt;cpuid&gt;.steal</code>      | -  | > 2% over 60 second period  |
| <b>CPU I/O Wait</b><br>This monitors CPU I/O wait time. High CPU wait times may indicate CPUs waiting on disk access.   | <code>cpu.&lt;cpuid&gt;.wait</code>       | > 30 for more than 5 min   | > 50 for more than 10 min   |



| Name/Description  | Statistics/Formula   | Warning Threshold   | Major Threshold   |
|---|--|---|---|
| <b>Memory utilization</b><br>Memory is a system resource, which needs to be less than 80%. The swap threshold has been reduced for CPS, and swapping should occur when the system resources are exhausted and memory utilization hits 99%.  | $\left( \frac{\text{memory.used}}{\text{memory.used} + \text{memory.free} + \text{memory.buffered} + \text{memory.cached}} \right) * 100$                                    | > 70% utilization over 60 second period                               | > 80% utilization over 60 second period                             |
| <b>Disk Utilization</b><br>Disk storage is a critical system resource, and when file system utilization exceeds 90% utilization the system can become less efficient. When the file system utilization hits 100%, then application can stop functioning.                                  | $\text{df.<fs>.df\_complex.free} - \text{df.<fs>.df\_complex.used}$  | > 80% utilization   | > 90% utilization   |
| <b>Session Store utilization</b><br>This KPI monitors the amount of database storage available. The data is evenly distributed across all shards, so any specific shard will have the same utilization rate as all shards.  | $\text{var-data-sessions\_1-free} - \text{var-data-sessions\_1-used}$  | > 70% utilization   | > More than 80% utilization   |
| <b>In Queue</b><br>These statistics monitors how long a message waits in the application queue, waiting to be serviced. The value should be 0 all the time. Non-zero values indicate the application is too slow, short of resources, or overwhelmed.                                     | $\text{node1.messages.in\_q}^*.avg$  | -   | More than 1 ms over 60 seconds                                      |
| <b>Diameter 3xxx errors</b><br>Diameter Too Busy 3xxx message indicate that the PCRF is overwhelmed, or responding too slowly. This can be related to In Queue issues, system resources, database problems, network latency, or issues with SPR or other external nodes in the call flow. | $\frac{\text{messages.e2e\_}\_3xxx.success}{\text{messages.e2e\_}\_2001.success}$ (and exclude the late statistics)<br>as a percentage of *.node*.messages.e2e_*2001.success | > 0.5% of *.node*.messages.e2e_*2001.success<br>Over 30 minute period | > 1% of *.node*.messages.e2e_*2001.success<br>Over 30 minute period |

| Name/Description  | Statistics/Formula   | Warning Threshold  | Major Threshold  |
|---|--|--|--|
| Diameter 5xxx errors<br>Session Not Found and other Diameter 5xxx errors indicate a critical problem with the ability to process the incoming diameter message. This can be related to exhausted PCRF system resources, invalid session id or bad message structure, length, or content, or even database corruption. | $\frac{\text{messages.e2e\_}^*\text{5xxx.success}}{(\text{and exclude the late statistics})}$<br>as a percentage of $\frac{\text{*.node*.messages.e2e\_}^*\text{2001.success}}{\text{e2e\_}^*\text{2001.success}}$ | > 0.5% of $\frac{\text{*.node*.messages.e2e\_}^*\text{2001.success}}{\text{e2e\_}^*\text{2001.success}}$<br>Over 5 minute period | > 1% of $\frac{\text{*.node*.messages.e2e\_}^*\text{2001.success}}{\text{e2e\_}^*\text{2001.success}}$<br>Over 5 minute period |
| Diameter Message Response Time  | -  | > 100 ms for more than 30 minutes  | > 300 ms for more than 15 minutes  |
| Active Session Count  | set_session_count_total.<br>records  | >80% of the lessor of the dimensioned or licensed capacity for more than 1 hour<br>or<br>= 0 for more than 5 minutes             | >80% of the lessor of the dimensioned or licensed capacity for more than 10 minutes<br>or<br>= 0 for more than 10 minutes      |
| Policy Execution Count (Internal TPS)   | -  | > 80% of the lessor of the dimensioned TPS capacity for more than 1 hour<br>or<br>= 0 for more than 5 minutes                    | > 80% of the lessor of the dimensioned TPS capacity for more than 10 minutes<br>or<br>= 0 for more than 10 minutes             |
| Policy Errors   | -  | > 0  | > 20 within 5 minutes  |

| Name/Description   | Statistics/Formula   | Warning Threshold   | Major Threshold   |
|--|--|---|---|
| Dedicated Bearer Errors  | $\frac{\text{node1.counters.}<domain>_{[realm\_]} \text{Gx\_bearer\_setup\_qci\_}<qci>_{[fail\_}<failure-code> \text{.qns\_count}}{\text{node1.counters.}<domain>_{[realm\_]} \text{Gx\_bearer\_setup\_qci\_}<qci> \text{.qns\_count}}$ as a percentage of node1.counters. | > .1  | > .5  |
| % of failed VoLTE calls due to resource allocation<br><br>This KPI monitors failed VoLTE calls due to resource allocation errors on the PCEF. A spike in this measurement does not indicate a CPS issue, but may flag an issue in the mobile network that should be investigated.                                | -  | > .1  | > .5  |
| % of Messages dropped due to SLA timeout<br><br>Messages dropped due to SLA timeouts indicate that the PCRF is overwhelmed, or responding too slowly. This can be related to In Queue issues, system resources, database problems, network latency, or issues with SPR or other external nodes in the call flow. | $\frac{\text{node1.counters.}[realm\_]*_{drop.qns\_count}}{\text{node1.counters.}[realm\_]*_{drop.qns\_count} + \text{node1.counters.}[realm\_]*_{success.qns\_count}}$ as a percentage of *.node*.messages.   | $> 0.5\% \text{ of } *.node*.messages. \text{ e2e\_} *2001.success$ | $> 1\% \text{ of } *.node*.messages. \text{ e2e\_} *2001.success$ |

## Session Monitoring KPIs

The following KPIs enable you to monitor CPS session operation volumes, error counts and other useful statistics.



### Note

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

Table 4: Session Monitoring KPIs

| Category          | Name/Description                                    | Statistics/Formula   | Availability/<br>Node |
|-------------------|---|--|-----------------------|
| Session Operation | Errored session creation count                      | node1.actions.CreateEntry.<br>qns_stat.error                     | Policy Server (qns)   |
| Session Operation | Successful session creation count                   | node1.actions.CreateEntry.<br>qns_stat.success                   | Policy Server (qns)   |
| Session Operation | Total milliseconds of successful session creations  | node1.actions.CreateEntry.<br>qns_stat.total_time_in_ms          | Policy Server (qns)   |
| Session Operation | Errored session deletion count                      | node1.actions.DeleteEntry.<br>qns_stat.error                     | Policy Server (qns)   |
| Session Operation | Successful session deletion count                   | node1.actions.DeleteEntry.<br>qns_stat.success                   | Policy Server (qns)   |
| Session Operation | Total milliseconds of successful session deletions  | node1.actions.DeleteEntry.<br>qns_stat.total_time_in_ms          | Policy Server (qns)   |
| Session Operation | Errored session retrieval count                     | node1.actions.<br>GetSessionAction.<br>qns_stat.error            | Policy Server (qns)   |
| Session Operation | Successful session retrieval count                  | node1.actions.<br>GetSessionAction.<br>qns_stat.success          | Policy Server (qns)   |
| Session Operation | Total milliseconds of successful session retrievals | node1.actions.<br>GetSessionAction.<br>qns_stat.total_time_in_ms | Policy Server (qns)   |
| Session Operation | Errored session update count                        | node1.actions.UpdateEntry.<br>qns_stat.error                     | Policy Server (qns)   |
| Session Operation | Successful session update count                     | node1.actions.UpdateEntry.<br>qns_stat.success                   | Policy Server (qns)   |
| Session Operation | Total milliseconds of successful session updates    | node1.actions.UpdateEntry.<br>qns_stat.total_time_in_ms          | Policy Server (qns)   |

| Category                 | Name/Description  | Statistics/Formula   | Availability/<br>Node |
|--------------------------|---|--|-----------------------|
| Internal Messages        | Errored timer messages  | node1.messages.<br>TimerExpired.<br>qns_stat.error                                   | Policy Server (qns)   |
| Internal Messages        | Successful timer messages   | node1.messages.<br>TimerExpired.<br>qns_stat.success                                 | Policy Server (qns)   |
| Session                  | Gauge count of lock percentage  | <set_name>.<br>lock.percent  | sessionmgr            |
| Session                  | Gauge count of delete operations  | <set_name>.<br>op_delete.gauge   | sessionmgr            |
| Session                  | Gauge count of insert operations  | <set_name>.<br>op_insert.gauge   | sessionmgr            |
| Session                  | Gauge count of update operations  | <set_name>.<br>op_update.gauge   | sessionmgr            |
| Secondary Key Operations | Per ring count of failed lookup for primary key using the secondary key in cache ring   | node1.counters.skcache_ring<br><1/2>_cache_miss.<br>qns_count                        | Policy Server (qns)   |
| Session Type Count       | Count of session types (GX_TGPP/RX_TGPP/SY_PRIME/SD_V11 ... etc) in active session DB partition per admin set   | <setid>.set_<br><set number of admin db><br>_session_type_<br><session_type>.records | sessionmgr            |
| Session Count            | Count of sessions in all active session DB partitions<br><br>Threshold: > 80% of dimensioned or licensed capacity for more than 1 hour, or = 0 (zero) for more than 5 minutes | set_session_count_<br>total.records  | Policy Server (qns)   |

## Diameter Monitoring KPIs

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



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

**Table 5: Diameter Monitoring KPIs**

| Appld/<br>Monitoring Area | Category            | Statistic  | Description  | Availability/<br>Node |
|---------------------------|---------------------|--|--|-----------------------|
| Gx/F                      | Diameter Round Trip | node[x].messages.e2e_<br>_<domain>_[realm_]<br>Gx_CCR-I_2001.<br>qns_stat.success              | Success message count for return code 2001                               | Policy Director       |
| Gx/F                      | Diameter Round Trip | node[x].messages.e2e_<br>_<domain>_[realm_]<br>Gx_CCR-I_2001.<br>qns_stat.total<br>_time_in_ms | Total milliseconds of successful messages with return code matching 2001 | Policy Director       |
| Gx/F                      | Diameter Round Trip | node[x].messages.e2e_<br>_<domain>_[realm_]<br>Gx_CCR-I_3xxx.<br>qns_stat.success              | Success count of messages with return code matching 3XXX                 | Policy Director       |
| Gx/F                      | Diameter Round Trip | node[x].messages.e2e_<br>_<domain>_[realm_]<br>Gx_CCR-I_4xxx.<br>qns_stat.success              | Success count of messages with return code matching 4XXX                 | Policy Director       |
| Gx/F                      | Diameter Round Trip | node[x].messages.e2e_<br>_<domain>_[realm_]<br>Gx_CCR-I_5xxx.<br>qns_stat.success              | Success count of messages with return code matching 5XXX                 | Policy Director       |

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

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



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

| Appld/<br>Monitoring Area | Category             | Statistic   | Description  | Availability/<br>Node |
|---------------------------|----------------------|---|--|-----------------------|
| Gx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Gx_CCR-T.qns_count   | Count of messages successful sent to the policy engine                   | Policy Server (qns)   |
| Gx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Gx_RAR_2001.<br>qns_stat.success              | Success message count for return code 2001                               | Policy Director       |
| Gx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Gx_RAR_2001.<br>qns_stat.total_<br>time_in_ms | Total milliseconds of successful messages with return code matching 2001 | Policy Director       |
| Gx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Gx_RAR_3xxx.<br>qns_stat.success              | Success count of messages with return code matching 3XXX                 | Policy Director       |
| Gx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Gx_RAR_4xxx.<br>qns_stat.success              | Success count of messages with return code matching 4XXX                 | Policy Director       |
| Gx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Gx_RAR_5xxx.<br>qns_stat.success              | Success count of messages with return code matching 5XXX                 | Policy Director       |

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

| Appld/<br>Monitoring Area | Category             | Statistic   | Description  | Availability/<br>Node |
|---------------------------|----------------------|---|--|-----------------------|
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_AAR_3xxx.<br>qns_stat.success    | Success count of<br>messages with<br>return code<br>matching 3XXX      | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_AAR_4xxx.<br>qns_stat.success    | Success count of<br>messages with<br>return code<br>matching 4XXX      | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_AAR_5xxx.<br>qns_stat.success    | Success count of<br>messages with<br>return code<br>matching 5XXX      | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_AAR_timeout.<br>qns_stat.success | Success count of<br>messages with<br>return code<br>matching 5XXX      | Policy Director       |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_RAA.qns_count                                       | Count of messages<br>successful sent to<br>the Policy Director<br>(LB) | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_AAR_drop.<br>qns_count                              | Count of messages<br>dropped due to<br>exceeding SLA                   | Policy Server (qns)   |

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

| Appld/<br>Monitoring Area | Category             | Statistic   | Description  | Availability/<br>Node |
|---------------------------|----------------------|---|--|-----------------------|
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_ASR_retry.<br>qns_count                    | Retry count for ASR message  | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_ASA_bypass.<br>qns_count                                      | Count of message that do not require processing by the policy engine     | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]Rx_ASA.<br>qns_count   | Count of messages successful sent to the policy engine                   | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_ASA_drop.<br>qns_count  | Count of messages dropped due to exceeding SLA                           | Policy Server (qns)   |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_RAR_2001.<br>qns_stat.success              | Success message count for return code 2001                               | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_RAR_2001.<br>qns_stat.total_<br>time_in_ms | Total milliseconds of successful messages with return code matching 2001 | Policy Director       |

| Appld/<br>Monitoring Area | Category             | Statistic  | Description  | Availability/<br>Node |
|---------------------------|----------------------|--|--|-----------------------|
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_RAR_3xxx.<br>qns_stat.success | Success count of messages with return code matching 3XXX             | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_RAR_4xxx.<br>qns_stat.success | Success count of messages with return code matching 4XXX             | Policy Director       |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_RAR_5xxx.<br>qns_stat.success | Success count of messages with return code matching 5XXX             | Policy Director       |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_RAA_bypass.<br>qns_count                         | Count of message that do not require processing by the policy engine | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_RAA.qns_count                                    | Count of messages successful sent to the policy engine               | Policy Server (qns)   |
| Rx/A                      | Diameter Input Queue | node1.counters.<br>[realm_]<br>Rx_RAA_drop.<br>qns_count                           | Count of messages dropped due to exceeding SLA                       | Policy Server (qns)   |
| Rx/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]<br>Rx_STR_2001.<br>qns_stat.success | Success message count for return code 2001                           | Policy Director       |

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



| Appld/<br>Monitoring Area | Category             | Statistic   | Description   | Availability/<br>Node |
|---------------------------|----------------------|---|---|-----------------------|
| Rx/A                      | Diameter Input Queue | node1.messages.<br>in_q_Rx_STR.<br>qns_stat.<br>total_time_in_ms                            | Total milliseconds of messages successfully sent to the policy engine                       | Policy Server (qns)   |
| Rx/D                      | Engine Message       | node1.messages.<br>diameter_Rx_STR.<br>qns_stat.success                                     | Success message count   | Policy Server (qns)   |
| Rx/D                      | Engine Message       | node1.messages.<br>diameter_Rx_STR.<br>qns_stat.<br>total_time_in_ms                        | Total milliseconds of successful messages   | Policy Server (qns)   |
| Rx/E                      | Diameter Input Queue | node1.counters.<br>[realm_]Rx_STA_2001.<br>qns_count  | Count of STA messages with result-code = 2001 sent successfully to the Policy Director (LB) | Policy Server (qns)   |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]Sy_SLR_2001.<br>qns_stat.success              | Success message count for return code 2001  | Policy Director       |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]Sy_SLR_2001.<br>qns_stat.<br>total_time_in_ms | Total milliseconds of successful messages with return code matching 2001                    | Policy Director       |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_]Sy_SLR_3xxx.<br>qns_stat.success              | Success count of messages with return code matching 3XXX                                    | Policy Director       |

| Appld/<br>Monitoring Area | Category             | Statistic   | Description   | Availability/<br>Node |
|---------------------------|----------------------|---|---|-----------------------|
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_] Sy_SLR_4xxx.<br>qns_stat.success | Success count of messages with return code matching 4XXX              | Policy Director       |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_] Sy_SLR_5xxx.<br>qns_stat.success | Success count of messages with return code matching 5XXX              | Policy Director       |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_SLR_bypass.<br>qns_count                         | Count of message that do not require processing by the policy engine  | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_SLR.qns_count                                    | Count of messages successful sent to the policy engine                | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_SLR_drop.qns_count                               | Count of messages dropped due to exceeding SLA                        | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.messages.<br>in_q_Sy_SLA.<br>qns_stat.success                             | Count of messages successfully sent to the policy engine              | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.messages.<br>in_q_Sy_SLA.<br>qns_stat.<br>total_time_in_ms                | Total milliseconds of messages successfully sent to the policy engine | Policy Server (qns)   |
| Sy/D                      | Engine Message       | node1.messages.<br>diameter_Sy_SLA.<br>qns_stat.success                         | Success message count   | Policy Server (qns)   |

| Appld/<br>Monitoring Area | Category               | Statistic  | Description  | Availability/<br>Node |
|---------------------------|------------------------|--|--|-----------------------|
| Sy/D                      | Engine Message         | node1.messages.<br>diameter_Sy_SLA.<br>qns_stat.<br>total_time_in_ms                             | Total milliseconds<br>of successful<br>messages                                      | Policy Server (qns)   |
| Sy/B                      | Diameter Action        | node1.actions.<br>send.diameter_<br>Sy_SLR.qns_stat.success                                      | Success actions<br>count   | Policy Server (qns)   |
| Sy/B                      | Diameter Action        | node1.actions.<br>send.diameter_<br>Sy_SLR.qns_stat.<br>total_time_in_ms                         | Total milliseconds<br>of successful actions  | Policy Server (qns)   |
| Sy/F                      | Diameter Round<br>Trip | node[x].messages.<br>e2e_<domain>_<br>[realm_]_<br>Sy_SNR_2001.<br>qns_stat.success              | Success message<br>count for return code<br>2001                                     | Policy Director       |
| Sy/F                      | Diameter Round<br>Trip | node[x].messages.<br>e2e_<domain>_<br>[realm_]_<br>Sy_SNR_2001.<br>qns_stat.<br>total_time_in_ms | Total milliseconds<br>of successful<br>messages with<br>return code<br>matching 2001 | Policy Director       |
| Sy/F                      | Diameter Round<br>Trip | node[x].messages.<br>e2e_<domain>_<br>[realm_]_<br>Sy_SNR_3xxx.<br>qns_stat.success              | Success count of<br>messages with<br>return code<br>matching 3XXX                    | Policy Director       |
| Sy/F                      | Diameter Round<br>Trip | node[x].messages.<br>e2e_<domain>_<br>[realm_]_<br>Sy_SNR_4xxx.<br>qns_stat.success              | Success count of<br>messages with<br>return code<br>matching 4XXX                    | Policy Director       |

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

| Appld/<br>Monitoring Area | Category             | Statistic   | Description  | Availability/<br>Node |
|---------------------------|----------------------|---|--|-----------------------|
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_] Sy_STR_3xxx.<br>qns_stat.success | Success count of messages with return code matching 3XXX             | Policy Director       |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_] Sy_STR_4xxx.<br>qns_stat.success | Success count of messages with return code matching 4XXX             | Policy Director       |
| Sy/F                      | Diameter Round Trip  | node[x].messages.<br>e2e_<domain>_<br>[realm_] Sy_STR_5xxx.<br>qns_stat.success | Success count of messages with return code matching 5XXX             | Policy Director       |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_STA_bypass.<br>qns_count                         | Count of message that do not require processing by the policy engine | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_STA.qns_count                                    | Count of messages successful sent to the policy engine               | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.counters.<br>[realm_] Sy_STA_drop.<br>qns_count                           | Count of messages dropped due to exceeding SLA                       | Policy Server (qns)   |
| Sy/A                      | Diameter Input Queue | node1.messages.<br>in_q_Sy_STA.<br>qns_stat.success                             | Count of messages successfully sent to the policy engine             | Policy Server (qns)   |

| Appld/<br>Monitoring Area | Category              | Statistic  | Description   | Availability/<br>Node |
|---------------------------|-----------------------|--|---|-----------------------|
| Sy/A                      | Diameter Input Queue  | node1.messages.<br>in_q_Sy_STA.<br>qns_stat.total_<br>time_in_ms         | Total milliseconds of messages successfully sent to the policy engine | Policy Server (qns)   |
| Sy/D                      | Engine Message        | node1.messages.<br>diameter_Sy_STA.<br>qns_stat.success                  | Success message count   | Policy Server (qns)   |
| Sy/D                      | Engine Message        | node1.messages.<br>diameter_Sy_STA.<br>qns_stat.<br>total_time_in_ms     | Total milliseconds of successful messages                             | Policy Server (qns)   |
| Sy/B                      | Diameter Action       | node1.actions.send.<br>diameter_Sy_STR.<br>qns_stat.success              | Success actions count   | Policy Server (qns)   |
| Sy/B                      | Diameter Action       | node1.actions.send.<br>diameter_<br>Sy_STR.qns_stat.<br>total_time_in_ms | Total milliseconds of successful actions                              | Policy Server (qns)   |
| Sy/E                      | Diameter Output Queue | node1.counters.<br>[realm_]_<br>Sy_STR.qns_count                         | Count of messages successfully sent to the Policy Director (LB)       | Policy Server (qns)   |

## Database Fragmentation Monitoring KPIs

CPS supports KPIs to monitor MongoDB level fragmentation using Grafana and generate an SNMP alarm when MongoDB fragment percentage exceeds a specified value.

The following databases are monitored for fragmentation statistics:

**Table 6: List of Databases for Monitoring Fragmentation**

| Replica Set Type | Database Name                     | Collections   | Comment          |
|------------------|-----------------------------------|---------------|------------------|
| Session          | session_cache, session_cache_2... | session       | Session and UDC  |
| Session          | sk_cache, sk_cache_2...           | secondary_key | Secondary Key DB |

| Replica Set Type | Database Name                  | Collections | Comment               |
|------------------|--------------------------------|-------------|-----------------------|
| ADMIN or SESSION | diameter                       | endpoints   | -                     |
| SPR              | spr                            | subscriber  | -                     |
| BALANCE          | balance_mgmt,balance_mgmt_2... | account     | Covers all the shards |

### Fragmentation KPI

- The following query must be configured in Grafana dashboard to view different statistics:

#### Fragmentation Percent

```
cisco.quantum.qps.<hostName>.<replicaSetName>_<hostName>_<port>_<dbName>.mongo_frag.fragPercent
```

#### File Size

```
cisco.quantum.qps.<hostName>.<replicaSetName>_<hostName>_<port>_<dbName>.mongo_frag.fileSize
```

#### Index Size

```
cisco.quantum.qps.<hostName>.<replicaSetName>_<hostName>_<port>_<dbName>.mongo_frag.indexSize
```

#### Data Size

```
cisco.quantum.qps.<hostName>.<replicaSetName>_<hostName>_<port>_<dbName>.mongo_frag.dataSize
```

#### Derived File Size

```
cisco.quantum.qps.<hostName>.<replicaSetName>_<hostName>_<port>_<dbName>.mongo_frag.derivedFileSize
```

Statistics once gathered are published to Grafana via collectd.

- Bulk statistics CSV files are created in /var/broadhop/stats/ directory on pcrfclient01 host. Statistics from agg-stats.0.csv are:

```
[root@pcrfclient01 ~]# grep -Eo ".*sessionmgr01.*?set01_.*?27717_session_cache\.mongo.*"
/var/broadhop/stats/agg-stats.0.csv
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.indexSize,429665152.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.storageSize,5336334224.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.fileSize,5886181376.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.documentCountInColl,646142.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.dataSize,5282857696.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.derivedFileSize,5349310464.0
2020-01-27T05:29:26.158+0000,G,sessionmgr01.set01_sessionmgr01_27717_session_cache.mongo_frag.fragPercent,3.0
```

- The script generates the following log file for troubleshooting purpose on sessionmgr VMs:

```
/var/log/mongodb-frag-check.log
```

- The script generates a replica set wise temporary csv files local to host. The files are of the following pattern:

```
/var/tmp/<replicaSetName>_frag_stats.csv
```

- The statistics for session\_cache, sk\_cache, diameter, and SPR databases (using primary members) can be checked using the following command:

```
diagnostics.sh --get_frag_status
```



**Note** The `diagnostics.sh --get_frag_status` like `diagnostics.sh --get_session_shard_health` is supported with root user only for multi-cluster HA and GR setups.

### Alarm Generation

- The script reads `/var/tmp/<replicaSetName>_frag_stats.csv` files to decide whether to send up or down alarm.
- The following trap generation script also logs the data to log file:

```
/var/log/broadhop/scripts/gen-frag-trap.log
```

## Configure Custom Database Fragmentation Threshold Percentage

The `/etc/collectd.d/dbMonitorList.cfg` file present on sessionmgr VMs contains the list of databases and their corresponding fragmentation threshold percentage values. By default, the fragmentation threshold value is 40 %. If a different threshold value needs to be updated, perform the following steps:

**Step 1** Update the `Configuration.csv` file with the required database name and their corresponding threshold percentage value. The format to provide the custom threshold percentage value is as follows (where, XX is numeric value of percentage).

```
session_cache,XX,
sk_cache,XX,
diameter,XX,
spr,XX,
balance_mgmt,XX,
```

**Step 2** Execute the following commands to update the `/etc/collectd.d/dbMonitorList.cfg` file so that it has the new threshold values from `Configuration.csv` file:

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

## Example CPS Statistics

### Sample CSV Files

The following list is a sample of the file names created in the `/var/broadhop/stats` directory on the `pcrfclient01` VM.

```
[root@pcrfclient01 stats]# pwd
/var/broadhop/stats
[root@pcrfclient01 stats]# ls
bulk-pcrfclient01-201510131350.csv
bulk-pcrfclient01-201510131355.csv
bulk-pcrfclient01-201510131400.csv
bulk-pcrfclient01-201510131405.csv
```



```

bulk-pcrfclient01-201510131410.csv
bulk-pcrfclient01-201510131415.csv
bulk-pcrfclient01-201510131420.csv
bulk-pcrfclient01-201510131425.csv
bulk-pcrfclient01-201510131430.csv
bulk-pcrfclient01-201510131435.csv
bulk-pcrfclient01-201510131440.csv
bulk-pcrfclient01-201510131445.csv
bulk-pcrfclient01-201510131450.csv
bulk-pcrfclient01-201510131455.csv
bulk-pcrfclient01-201510131500.csv
bulk-pcrfclient01-201510131505.csv
bulk-pcrfclient01-201510131510.csv
bulk-pcrfclient01-201510131515.csv
bulk-pcrfclient01-201510131520.csv
bulk-pcrfclient01-201510131525.csv
bulk-pcrfclient01-201510131530.csv
bulk-pcrfclient01-201510131535.csv
bulk-pcrfclient01-201510131540.csv
bulk-pcrfclient01-201510131545.csv
bulk-pcrfclient01-201510131550.csv
bulk-pcrfclient01-201510131555.csv
bulk-pcrfclient01-201510131600.csv
bulk-pcrfclient01-201510131605.csv
bulk-pcrfclient01-201510131610.csv
bulk-pcrfclient01-201510131615.csv
bulk-pcrfclient01-201510131620.csv
bulk-pcrfclient01-201510131625.csv
bulk-pcrfclient01-201510131630.csv

```

## Sample Output

C,<VM\_name>,node1.actions.send.diameter\_Gx\_CCA-I.qns\_stat.success,19

where, the <VM\_Name> indicates which VM the statistics has been collected on.

A sample bulk statistics .csv file is shown below:

```

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

```

