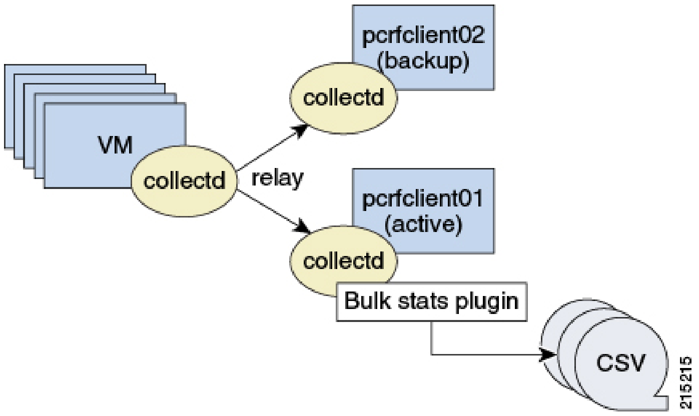
CPS Statistics

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 +/- 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

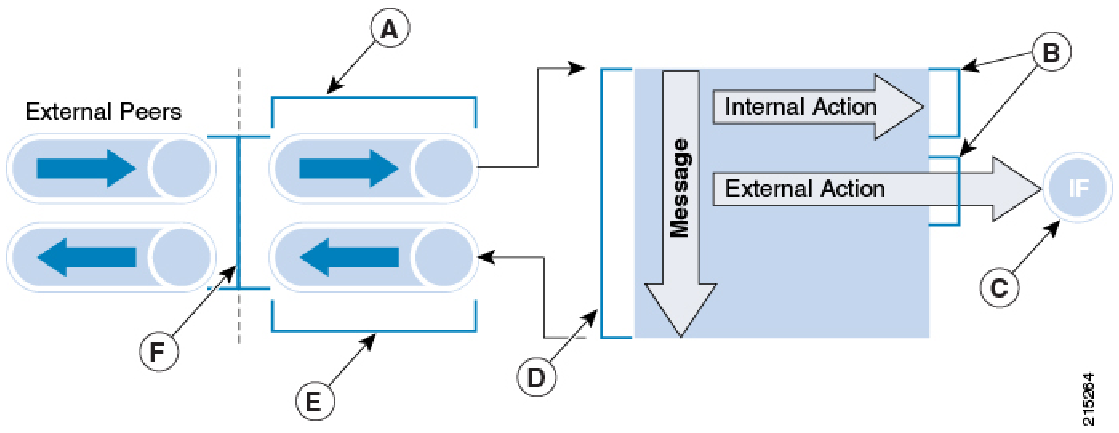


Table 1: Measurement Legend

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 integer, which can increase or decrease, but can never exceed a maximum value, nor fall below a minimum value. The maximum value cannot be greater than 2^32-1 (4294967295 decimal), and 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

MOG API Statistics

API statistics are defined based on five categories: Bearer Count, Tenant Onboarding Count, Subscriber Onboarding Count, Authentication Count and Callback Response Statistics.

Default and Dedicated Bearer Counters

Counter for the number of default and dedicated bearers related to API requests.

Default and Dedicated Bearer Statistics

Provides the statistics for default and dedicated bearers related to API requests.

Tenant Onboarding Counters

Counter for the number of tenant onboarding related to API requests.

Tenant Onboarding Statistics

Provides the statistics for tenant onboarding related to API requests.

Subscriber Onboarding Counters

Counter for the number of subscriber onboarding related to API requests.

Subscriber Onboarding Statistics

Provide the statistics for subscriber onboarding related to API requests.

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)

Error Statistics	Description
_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 pcrfclient01 and pcrfclient02 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 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
```

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
CPU Utilization 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.	100 - cpu.<*cpuid*>.idle	> 60% utilization over 60 second period (assuming that idle is less than 40%)	> 80% utilization over 60 second period (assuming idle is less than 20%)
CPU Steal 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.	cpu.<*cpuid*>.steal	-	> 2% over 60 second period
CPU I/O Wait This monitors CPU I/O wait time. High CPU wait times may indicate CPUs waiting on disk access.	cpu.<*cpuid*>.wait	> 30 for more than 5 min	> 50 for more than 10 min
Memory utilization 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%.	memory.free – memory.used	> 70% utilization over 60 second period	> 80% utilization over 60 second period
Disk Utilization 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.	df.<fs>.df_complex.free - df.<fs>.df_complex.used	> 80% utilization	> 90% utilization

Name/Description	Statistics/Formula	Warning Threshold	Major Threshold
Session Store utilization 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.	var-data-sessions_1-free - var-data-sessions_1-used	> 70% utilization	> More than 80% utilization
In Queue 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.	node1.messages.in_q*.avg	-	More than 1 ms over 60 seconds
Database lock This KPI monitors if database locking is excessive in a platform. Database locking is normal and expected, but the locks should be quick and a very low percentage. Values higher than 20% indicate problems with the database operation or the system resources associated with the database such as disk or memory.	*lock.percent	> 15% lock	> 20% lock
Diameter 3xxx errors 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.	messages.e2e_*_ 3xxx.success (and exclude the late statistics) as a percentage of *.node*.messages. e2e_*2001.success	> 0.5% of *.node*.messages. e2e_*2001.success Over 30 minute period	> 1% of *.node*.messages. e2e_*2001.success Over 30 minute period
Diameter 5xxx errors 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.	messages.e2e_*_ 5xxx.success (and exclude the late statistics) as a percentage of *.node*.messages. e2e_*2001.success	> 0.5% of *.node*.messages. e2e_*2001.success Over 5 minute period	> 1% of *.node*.messages. e2e_*2001.success Over 5 minute period

Name/Description	**Statistics/Formula**	**Warning Threshold**	**Major Threshold**
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. records	>80% of the lessor of the dimensioned or licensed capacity for more than 1 hour or = 0 for more than 5 minutes	>80% of the lessor of the dimensioned or licensed capacity for more than 10 minutes or = 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 or = 0 for more than 5 minutes	> 80% of the lessor of the dimensioned TPS capacity for more than 10 minutes or = 0 for more than 10 minutes
Policy Errors	-	> 0	> 20 within 5 minutes
Dedicated Bearer Errors	node1.counters.<*domain*> _[realm_] Gx_bearer_setup_qci_<*qci*> _fail_<*failure-code*> .qns_count as a percentage of node1.counters. <*domain*> _[realm_] Gx_bearer_setup_qci_<*qci*>. qns_count	> .1	> .5

Name/Description	Statistics/Formula	Warning Threshold	Major Threshold
% of failed VoLTE calls due to resource allocation 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 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.	node1.counters.[realm_]* _drop.qns_count as a percentage of *.node*.messages. e2e_*2001.success	> 0.5% of *.node*.messages. e2e_*2001.success	> 1% of *.node*.messages. 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/ Node
Session Operation	Errored session creation count	node1.actions.CreateEntry. qns_stat.error	Policy Server (qns)
Session Operation	Successful session creation count	node1.actions.CreateEntry. qns_stat.success	Policy Server (qns)
Session Operation	Total milliseconds of successful session creations	node1.actions.CreateEntry. qns_stat.total_time_in_ms	Policy Server (qns)

Category	Name/Description	Statistics/Formula	Availability/ Node
Session Operation	Errored session deletion count	node1.actions.DeleteEntry. qns_stat.error	Policy Server (qns)
Session Operation	Successful session deletion count	node1.actions.DeleteEntry. qns_stat.success	Policy Server (qns)
Session Operation	Total milliseconds of successful session deletions	node1.actions.DeleteEntry. qns_stat.total_time_in_ms	Policy Server (qns)
Session Operation	Errored session retrieval count	node1.actions. GetSessionAction. qns_stat.error	Policy Server (qns)
Session Operation	Successful session retrieval count	node1.actions. GetSessionAction. qns_stat.success	Policy Server (qns)
Session Operation	Total milliseconds of successful session retrievals	node1.actions. GetSessionAction. qns_stat.total_ time_in_ms	Policy Server (qns)
Session Operation	Errored session update count	node1.actions.UpdateEntry. qns_stat.error	Policy Server (qns)
Session Operation	Successful session update count	node1.actions.UpdateEntry. qns_stat.success	Policy Server (qns)
Session Operation	Total milliseconds of successful session updates	node1.actions.UpdateEntry. qns_stat.total_ time_in_ms	Policy Server (qns)
Internal Messages	Errored timer messages	node1.messages. TimerExpired. qns_stat.error	Policy Server (qns)
Internal Messages	Successful timer messages	node1.messages. TimerExpired. qns_stat.success	Policy Server (qns)

Category	Name/Description	Statistics/Formula	Availability/ Node
Session	Gauge count of lock percentage	<*set_name*>. lock.percent	sessionmgr
Session	Gauge count of delete operations	<*set_name*>. op_delete.gauge	sessionmgr
Session	Gauge count of insert operations	<*set_name*>. op_insert.gauge	sessionmgr
Session	Gauge count of update operations	<*set_name*>. 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 <*1*|*2*>_cache_miss. 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_ <*set number of admin db*> _session_type_ <*session_type*>.records	sessionmgr
Session Count	Count of sessions in all active session DB partitions Threshold: > 80% of dimensioned or licensed capacity for more than 1 hour, or = 0 (zero) for more than 5 minutes	set_session_count_ 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

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

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

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

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

AppId/ Monitoring Area	Category	Statistic	Description	Availability/ Node
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Gx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Gx_RAR_timeout. qns_stat.success	Success timeout count for RAR message	Policy Director

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

AppId/ Monitoring Area	Category	Statistic	Description	Availability/ Node
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_AAR_timeout. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_RAA.qns_count	Count of messages successful sent to the Policy Director (LB)	Policy Server (qns)
Rx/A	Diameter Input Queue	node1.counters. [realm_] Rx_AAR_drop. qns_count	Count of messages dropped due to exceeding SLA	Policy Server (qns)
Rx/E	Diameter Output Queue	node1.counters. [realm_] Rx_AAA_2001. qns_count	Count of AAA messages with result-code = 2001 sent successfully to the Policy Director (LB)	Policy Server (qns)

AppId/ Monitoring Area	Category	Statistic	Description	Availability/ Node
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_2001. qns_stat.total_ time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_5xxx. qns_stat.success	Success count of messages with return code matching 5XXX	Policy Director
Rx/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Rx_ASR_retry. qns_count	Retry count for ASR message	Policy Server (qns)

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

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

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

AppId/ Monitoring Area	Category	Statistic	Description	Availability/ Node
Rx/D	Engine Message	node1.messages. diameter_Rx_STR. qns_stat.success	Success message count	Policy Server (qns)
Rx/D	Engine Message	node1.messages. diameter_Rx_STR. qns_stat. total_time_in_ms	Total milliseconds of successful messages	Policy Server (qns)
Rx/E	Diameter Input Queue	node1.counters. [realm_] Rx_STA_2001. qns_count	Count of STA messages with result-code = 2001 sent successfully to the Policy Director (LB)	Policy Server (qns)
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_2001. qns_stat.success	Success message count for return code 2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_2001. qns_stat. total_time_in_ms	Total milliseconds of successful messages with return code matching 2001	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_3xxx. qns_stat.success	Success count of messages with return code matching 3XXX	Policy Director
Sy/F	Diameter Round Trip	node[x].messages. e2e_<domain>_ [realm_] Sy_SLR_4xxx. qns_stat.success	Success count of messages with return code matching 4XXX	Policy Director

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

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

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

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

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

Example CPS Statistics

Sample CSV Files

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